F:\git\java\mar3\filemonitor\target\PHPCoins_V1.0\PHPCoins_V1.0-0.doc

0:F:\git\coin\\okbase.net\PHPCoins_V1.0\act.php

```
<?php
class act{
  function trade_close($id){
    db("update trade set status=-1 where id=? and user_id=?",[$id,mid()]);
    jump(url('my',['show'=>'trade_record']));
  }

  function log_out(){
    unset($_SESSION['user_id']);
    jump();
  }

  function my(){
    html(['header','navbar','my_body','footer']);
    receive_btc();
  }

  function email_validate($key,$email){
    $u = user(['email'=>$email]);
    $key == md5($u['ctime']) && user_edit(['id'=>$u['id'],'email_validated'=>1]);
    msg('');
  }

  function blank(){
    html(['header','navbar',$_GET['show'],'footer']);
  }

  function change_uid($uid){
    if(DEV){
      $_SESSION['user_id'] = $uid;
      header('location:/');
    }
  }

  function index(){
    html(['header','navbar','index_body','footer']);
  }

  function sign_up($email,$password,$pin,$confirm_password,$confirm_pin){
    v_assert([!empty($email)?:'',
      v_email($email)?:'',
      strlen($password)>=6 ?:'6',
      strlen($pin)>=6 ?:'6',
      $confirm_password == $password ?:'',
      $confirm_pin == $pin ?:'',
      $password != $pin ?:'']);
    if($_SESSION['user_id'] = user_add(['email'=>$email,
'password'=>md5($password),
'ctime'=>time(),
'secret'=>google_auth_create_secret(),
'address'=>btc()->getnewaddress($email),
'pin'=>md5($pin)])){
      return ['jump'=>'/'];
    }
  }

  function sign_in($email,$password){
    if($_SESSION['user_id'] = user(['email'=>$email,'password'=>md5($password)])['id']){
      return ['jump'=>'/'];
    }else{
      return ['msg'=>''];
    }
  }

  function withdraw($amount,$name,$bank,$card,$auth){
    db_lock(['user','withdraw']);
    v_assert([!empty($amount)?:'',
      !empty($name)?:'',
      !empty($bank)?:'',
      !empty($card)?:'',
      _rmbval($amount)!=0?:'',
      $amount <= floatval(user_info()['rmb']) ?:'']);
    if(user_info()['auth_withdraw']){
      v_assert([google_auth_verify($auth)?:'']);
    }
    if(!withdraw_add(['user_id'=>$_SESSION['user_id'],
      'amount'=>$amount,
      'name'=>$name,
      'card'=>$card,
      'bank'=>$bank,
      'ctime'=>time()]) ||
       !db("update user set rmb=rmb-?,rmb_frozen=rmb_frozen+? where
id=?",[$amount,$amount,mid()],1)){
      db_rollback();
      error();
    }
    db_unlock();
    return ['msg'=>'','jump'=>'/?act=my'];
  }

  function transfer($address,$amount,$auth){
    db_lock(['user','transfer']);
    v_assert([!empty($address)?:'',
      !empty($amount)?:'',
      _btcval($amount)!=0?:'',
      $amount <= floatval(user_info()['btc'])?:'']);
    if(user_info()['auth_withdraw']){
      v_assert([!empty($auth)?:'',
google_auth_verify($auth)?:'']);
    }
    if(!transfer_add(['user_id'=>mid(),
      'address'=>$address,
      'amount'=>$amount,
      'ctime'=>time()])||
       !db("update user set btc=btc-?,btc_frozen=btc_frozen+? where id=?",
   [$amount,$amount,mid()],1)){
db_rollback();
error();
    }
    db_unlock();
    return ['msg'=>'','jump'=>'/'];
  }

  function buy_btc($price,$amount,$pin,$auth){
    db_lock(['user','trade']);
    v_assert([!empty(_rmbval($price))?:'',
      !empty(_btcval($amount))?:'',
      md5($pin)==user_info()['pin']?:'',
      $price*$amount<=floatval(user_info()['rmb'])?:'']);
    if(user_info()['auth_trade']){
      v_assert([google_auth_verify($auth)?:'']);
    }
    if(!db("update user set rmb=rmb-?,rmb_frozen=rmb_frozen+? where
id=?",[$price*$amount,$price*$amount,$_SESSION['user_id']],1)||
       !trade_add(['amount'=>$amount,
   'price'=>$price,
   'ctime'=>time(),
   'user_id'=>mid()])){
      db_rollback();
      error();
    }
    db_unlock();
    run_deal();
    return ['msg'=>'','jump'=>'/?act=my'];
  }

  function sell_btc($price,$amount,$pin,$auth){
    db_lock(['user','trade']);
    v_assert([_rmbval($price)!=0?:'',
      _btcval($amount)!=0?:'',
      md5($pin)==user_info()['pin']?:'',
      $amount<=floatval(user_info()['btc'])?:'']);
    if(user_info()['auth_trade']){
      v_assert([google_auth_verify($auth)?:'wrong auth']);
    }
    if(!db("update user set btc=btc-?,btc_frozen=btc_frozen+? where
id=?",[$amount,$amount,$_SESSION['user_id']])||
       !trade_add(['amount'=>$amount,
   'price'=>$price,
   'ctime'=>time(),
   'user_id'=>mid(),
   'type'=>1])){
db_rollback();
error();
    }
    db_unlock();
    run_deal();
    return ['msg'=>'','jump'=>'/?act=my'];
  }

  function change_password($old,$password,$confirm){
    v_assert([md5($old)==user_info()['password']?:'',
      strlen($password)>=6?:'6',
      $confirm==$password?:'',
      md5($password)!=user_info()['pin']?:'']);
    if(user_edit(['id'=>mid(),
  'password'=>md5($password)])){
      return ['msg'=>'','jump'=>'/?act=my'];
    }
  }

  function change_pin($old,$pin,$confirm){
    v_assert([md5($old)==user_info()['pin']?:'',
      strlen($pin)>=6?:'6',
      $confirm==$pin?:'',
      md5($pin)!=user_info()['password']?:'']);
    if(user_edit(['id'=>mid(),
  'pin'=>md5($pin)])){
      return ['msg'=>'','jump'=>'/?act=my'];
    }
  }

  function auth_config($auth,$code){
    $auth = $auth ?: [];
    v_assert([google_auth_verify($code)?:'']);
    if(user_edit(['id'=>mid(),
  'secret_installed'=>1,
  'auth_trade'=>in_array('trade',$auth)?1:0,
  'auth_withdraw'=>in_array('withdraw',$auth)?1:0])){
      return ['msg'=>'','jump'=>'/?act=my'];
    }
  }

  function change_profile($name){
    if(user_edit(['id'=>mid(),
  'name'=>$name])){
      return ['msg'=>''];
    }
  }

  function password_reset_send($email){
    v_assert([!!user(['email'=>$email])?:'']);
    send_mail($email,'','<a
href="'.$_SERVER['REQUEST_SCHEME'].'://'.$_SERVER['HTTP_HOST'].'/?act=blank&show=pa
ssword_reset&email='.$email.'&key='.md5(db("select * from user where email=?",[$email])-
>fetch()['password']).'"></a>');
    return ['msg'=>''];
  }

  function password_reset($email,$password,$pin,$confirm_password,$confirm_pin,$key){
    v_assert([strlen($password)>=6?:'6',
      $password==$confirm_password?:'',
      strlen($pin)>=6?:'6',
      $pin==$confirm_pin?:'',
      $password != $pin ?:'',
      md5(user(['email'=>$email])['password'])==$key?:'wrong key']);
    if(user_edit(['id'=>user(['email'=>$email])['id'],
  'secret_installed'=>0,
  'auth_trade'=>0,
  'auth_withdraw'=>0,
  'password'=>md5($password),
  'pin'=>md5($pin)])){
      $_SESSION['user_id']=user(['email'=>$email])['id'];
      return ['msg'=>'','jump'=>'/'];
    }
  }

  function email_validate_send(){
    send_mail(user_info()['email'],'','<a
href="'.$_SERVER['REQUEST_SCHEME'].'://'.$_SERVER['HTTP_HOST'].'/?act=email_validate&
email='.user_info()['email'].'&key='.md5(user_info()['ctime']).'"></a>');
    return ['msg'=>''];
  }

  function help(){
    html(['header','navbar','help','footer']);
  }
}

1:F:\git\coin\\okbase.net\PHPCoins_V1.0\admin.php
<?php
class admin{
  function do_sign_in($captcha,$password){
    if(!captcha($captcha)){
      return ['msg'=>''];
    }elseif(config('admin_password') != $password){
      return ['msg'=>''];
    }else{
      $_SESSION['admin'] = 1;
      return ['jump'=>url('index')];
    }
  }

  function do_transfer($id){
    transfer_edit(['id'=>$id,'status'=>1]);
    jump();
  }

  function do_withdraw($id){
    withdraw_edit(['id'=>$id,'status'=>1]);
    jump();
  }

  function help_del($id){
    help_del($id);
    jump();
  }

  function log_out(){
    unset($_SESSION['admin']);
    jump();
  }

  function help_write($title,$body,$sort){
    v_assert([!empty($title)?:'',
      !empty($body)?:'']);
    help_add(['title'=>$title,'body'=>$body,'sort'=>_intval($sort)]);
    return ['msg'=>'','jump'=>url('index',['show'=>'help'])];
  }

  function
setting($site_name,$text_logo,$smtp_host,$smtp_port,$smtp_user,$smtp_password,$btc_protoco
l,$btc_user,$btc_password,$btc_host,$btc_port){
    config('site_name',$site_name);
    config('text_logo',$text_logo);
    config('smtp_host',$smtp_host);
    config('smtp_port',$smtp_port);
    config('smtp_user',$smtp_user);
    config('smtp_password',$smtp_password);
    config('btc_protocol',$btc_protocol);
    config('btc_user',$btc_user);
    config('btc_password',$btc_password);
    config('btc_host',$btc_host);
    config('btc_port',$btc_port);
    return ['msg'=>''];
  }

  function do_recharge($email,$amount){
    v_assert([!!user(['email'=>$email])?:'',
      _rmbval($amount)>0?:'']);
    if(db("update user set rmb=rmb+? where email=?",[$amount,$email],1)
       &&
recharge_add(['user_id'=>user(['email'=>$email])['id'],'amount'=>$amount,'ctime'=>time()]))
      return ['msg'=>'','jump'=>url('index',['show'=>'recharge'])];
  }
}
```

2:F:\git\coin\\okbase.net\PHPCoins_V1.0\back\analytics.html

```
<div class="well">
  <p>:<?=db("select sum(rmb+rmb_frozen) as sum from user")->fetch()['sum']?></p>
  <p>:<?=db("select sum(btc+btc_frozen) as sum from user")->fetch()['sum']?></p>
  <p>:<?=db("select sum(rmb_frozen) as sum from user")->fetch()['sum']?></p>
  <p>:<?=db("select sum(btc_frozen) as sum from user")->fetch()['sum']?></p>
  <p>:<?=db("select sum(rmb) as sum from user")->fetch()['sum']?></p>
  <p>:<?=db("select sum(btc) as sum from user")->fetch()['sum']?></p>
</div>
```

3:F:\git\coin\\okbase.net\PHPCoins_V1.0\back\deal.html

```
<table>
  <thead>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
  </thead>
<? foreach(db_page("select buy.price buy_price,sell.price sell_price,buy.user_id buyer,sell.user_id
seller,deal.ctime ctime,deal.amount amount from deal inner join trade buy on
deal.buy_trade=buy.id inner join trade sell on deal.sell_trade=sell.id order by deal.ctime desc") as
$v):?>
  <tr>
    <td>
```

```
      <?=user_info($v['buyer'])['email']?>
    </td>
    <td>
      <?=user_info($v['seller'])['email']?>
    </td>
    <td>
      <?=$v['buy_price']?>
    </td>
    <td>
      <?=$v['sell_price']?>
    </td>
    <td>
      <?=$v['amount']?>
    </td>
    <td>
      <?=date_full($v['ctime'])?>
    </td>
  </tr>
<? endforeach?>
</table>
<?=$GLOBALS['pager']?>
```

4:F:\git\coin\\okbase.net\PHPCoins_V1.0\back\do_recharge.html

```
<form act="do_recharge">
  <?=@form_text(['label'=>':','id'=>'email','col'=>[3,9]])?>
  <?=@form_text(['label'=>':','id'=>'amount'])?>
  <?=@form_submit('')?>
</form>
```

5:F:\git\coin\\okbase.net\PHPCoins_V1.0\back\footer.html

```
  </body>
</html>
```

6:F:\git\coin\\okbase.net\PHPCoins_V1.0\back\header.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta charset="utf-8">
    <title><?=@$GLOBALS['title']?:config('site_name')?></title>
    <link href="/public/css/bootstrap.min.css" rel="stylesheet" media="screen">
    <link href="/public/css/btc.css" rel="stylesheet" media="screen">
    <script>
     var ADMIN = <?=ADMIN?>
    </script>
  </head>
  <body>
    <script src="/public/js/jquery.js"></script>
    <script src="/public/js/bootstrap.js"></script>
    <script src="/public/js/highstock.src.js"></script>
    <script src="/public/js/btc.js"></script>
    <script src="/public/js/respond.js"></script>
```

7:F:\git\coin\\okbase.net\PHPCoins_V1.0\back\help.html

```
<div well>
  <a href="<?=url(['show'=>'help_write'])?>" class="btn btn-default pull-right"></a>
</div>
<table>
  <thead>
    <th></th>
    <th></th>
    <th></th>
  </thead>
<? foreach(db("select * from help order by sort desc")->fetchAll() as $v):?>
  <tr>
    <td>
      <?=$v['title']?>
    </td>
    <td>
      <?=$v['sort']?>
    </td>
    <td>
      <a href="<?=url(['show'=>'help_read','id'=>$v['id']])?>"></a>
      <a href="<?=url('help_del',['id'=>$v['id']])?>"></a>
    </td>
  </tr>
<? endforeach?>
</table>
```

8:F:\git\coin\\okbase.net\PHPCoins_V1.0\back\help_read.html

```
<h1 class="text-center">
  <?=help(['id'=>$_REQUEST['id']])['title']?>
```

```
</h1>
<p>
  <?=help(['id'=>$_REQUEST['id']])['body']?>
</p>
```

9:F:\git\coin\\okbase.net\PHPCoins_V1.0\back\help_write.html

```
<form act="help_write">
<?=@form_text(['label'=>':','id'=>'title','col'=>[3,9]])?>
<?=@form_textarea(['label'=>':','id'=>'body'])?>
<?=@form_text(['label'=>':','id'=>'sort','value'=>0])?>
<?=@form_submit('')?>
</form>
```

10:F:\git\coin\\okbase.net\PHPCoins_V1.0\back\index.html

```
<? html('header')?>
<div class="container">
  <div class="row">
    <div class="col-md-2" style="margin-top:50px">
      <? html('left')?>
    </div>
    <div class="col-md-10" style="margin-top:50px">
      <? html(@$_GET['show']?:'analytics')?>
    </div>
  </div>
</div>
<? html('footer')?>
```

11:F:\git\coin\\okbase.net\PHPCoins_V1.0\back\left.html

```
<div class="list-group">
  <a class="list-group-item" href="<?=url(['show'=>'setting'])?>"></a>
  <a class="list-group-item" href="<?=url(['show'=>'trade'])?>"></a>
  <a class="list-group-item" href="<?=url(['show'=>'deal'])?>"></a>
  <a class="list-group-item" href="<?=url(['show'=>'recharge'])?>"></a>
  <a class="list-group-item" href="<?=url(['show'=>'withdraw'])?>"></a>
  <a class="list-group-item" href="<?=url(['show'=>'transfer'])?>"></a>
  <a class="list-group-item" href="<?=url(['show'=>'help'])?>"></a>
  <a class="list-group-item" href="<?=url(['show'=>'user'])?>"></a>
  <a class="list-group-item" href="<?=url(['show'=>'do_recharge'])?>"></a>
  <a class="list-group-item" href="/"></a>
  <a class="list-group-item" href="<?=url('log_out')?>"></a>
</div>
```

12:F:\git\coin\\okbase.net\PHPCoins_V1.0\back\recharge.html

```
<table>
  <thead>
    <th></th>
    <th></th>
    <th></th>
  </thead>
<? foreach(db_page("select * from recharge order by ctime desc") as $v):?>
  <tr>
    <td>
      <?=user_info($v['user_id'])['email']?>
    </td>
    <td>
      <?=$v['amount']?>
    </td>
    <td>
      <?=date_full($v['ctime'])?>
    </td>
  </tr>
<? endforeach?>
</table>
<?=$GLOBALS['pager']?>
```

13:F:\git\coin\\okbase.net\PHPCoins_V1.0\back\setting.html

```
<form act="setting">
  <?=@form_text(['label'=>':','id'=>'site_name','value'=>config('site_name'),'col'=>[3,9]])?>
  <?=@form_text(['label'=>'LOGO:','id'=>'text_logo','value'=>config('text_logo')])?>
  <?=@form_text(['label'=>'Smtp :','id'=>'smtp_host','value'=>config('smtp_host')])?>
  <?=@form_text(['label'=>'Smtp :','id'=>'smtp_port','value'=>config('smtp_port')])?>
  <?=@form_text(['label'=>'Smtp :','id'=>'smtp_user','value'=>config('smtp_user')])?>
  <?=@form_text(['label'=>'Smtp :','id'=>'smtp_password','value'=>config('smtp_password')])?>
  <?=@form_text(['label'=>'Btc :','id'=>'btc_protocol','value'=>config('btc_protocol')])?>
  <?=@form_text(['label'=>'Btc :','id'=>'btc_host','value'=>config('btc_host')])?>
  <?=@form_text(['label'=>'Btc :','id'=>'btc_port','value'=>config('btc_port')])?>
  <?=@form_text(['label'=>'Btc :','id'=>'btc_user','value'=>config('btc_user')])?>
  <?=@form_text(['label'=>'Btc :','id'=>'btc_password','value'=>config('btc_password')])?>
  <?=@form_submit('')?>
</form>
```

14:F:\git\coin\\okbase.net\PHPCoins_V1.0\back\sign_in.html

```
<?=html('header')?>
<div class="container" style="margin-top:200px;">
  <div class="row">
    <div class="col-md-4 col-md-offset-4">
      <div class="well clearfix">
<form act="do_sign_in">
  <?=@form_password(['label'=>':','id'=>'password','col'=>[3,9]])?>
  <?=@form_text(['label'=>':','id'=>'captcha'])?>
  <?=@form_captcha()?>
  <?=@form_submit('')?>
</form>
      </div>
    </div>
  </div>
</div>
<?=html('footer')?>
```

15:F:\git\coin\\okbase.net\PHPCoins_V1.0\back\trade.html

```
<table>
  <thead>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
  </thead>
<? foreach(db_page("select * from trade order by ctime desc") as $v):?>
  <tr>
    <td>
      <?=user_info($v['user_id'])['email']?>
    </td>
    <td>
```

```
      <?=$v['type']==0?'':''?>
    </td>
    <td>
      <?=$v['price']?>
    </td>
    <td>
      <?=$v['amount']?>
    </td>
    <td>
      <?=$v['deal']?>
    </td>
    <td>
      <?=$v['amount']-$v['deal']?>
    </td>
    <td>
      <?=$v['status']==0?'':($v['status']==1?'':'')?>
    </td>
    <td>
      <?=date_full($v['ctime'])?>
    </td>
  </tr>
<? endforeach?>
</table>
<?=$GLOBALS['pager']?>

16:F:\git\coin\\okbase.net\PHPCoins_V1.0\back\transfer.html
<table>
  <thead>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
  </thead>
<? foreach(db_page("select * from transfer order by ctime desc") as $v):?>
  <tr>
    <td>
      <?=user_info($v['user_id'])['email']?>
    </td>
    <td>
      <?=$v['address']?>
    </td>
    <td>
      <?=$v['amount']?>
    </td>
    <td>
      <?=date_full($v['ctime'])?>
    </td>
    <td>
      <?=$v['status']==0?'':''?>
    </td>
    <td>
      <?=$v['status']==0?'<a href="'.url('do_transfer',['id'=>$v['id']]).'"></a>':''?>
    </td>
  </tr>
<? endforeach?>
</table>
<?=$GLOBALS['pager']?>
```

17:F:\git\coin\\okbase.net\PHPCoins_V1.0\back\user.html

```
<table>
  <thead>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
  </thead>
<? foreach(db_page("select * from user order by ctime desc") as $v):?>
  <tr>
    <td>
      <?=$v['email']?>
    </td>
    <td>
      <?=$v['name']?>
    </td>
    <td>
      <?=$v['address']?>
    </td>
    <td>
      <?=$v['rmb']?>
    </td>
    <td>
      <?=$v['rmb_frozen']?>
    </td>
    <td>
      <?=$v['btc']?>
    </td>
    <td>
      <?=$v['btc_frozen']?>
    </td>
    <td>
      <?=$v['received']?>
    </td>
    <td>
      <?=date_full($v['ctime'])?>
    </td>
  </tr>
<? endforeach?>
</table>
<?=$GLOBALS['pager']?>
```

18:F:\git\coin\\okbase.net\PHPCoins_V1.0\back\withdraw.html

```
<table>
  <thead>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
  </thead>
<? foreach(db_page("select * from withdraw order by ctime desc") as $v):?>
  <tr>
    <td>
      <?=user_info($v['user_id'])['email']?>
    </td>
    <td>
      <?=$v['amount']?>
    </td>
    <td>
      <?=$v['bank']?>
    </td>
    <td>
      <?=$v['card']?>
    </td>
    <td>
      <?=$v['name']?>
    </td>
    <td>
      <?=date_full($v['ctime'])?>
    </td>
    <td>
      <?=$v['status']==0?'':''?>
    </td>
    <td>
      <?=$v['status']==0?'<a href="'.url('do_withdraw',['id'=>$v['id']]).'"></a>':''?>
    </td>
  </tr>
<? endforeach?>
</table>
<?=$GLOBALS['pager']?>
```

19:F:\git\coin\\okbase.net\PHPCoins_V1.0\config.php

```
<?php
      return array (
  'admin_password' => 'aaaaaa',
  'db_user' => 'root',
  'db_password' => '',
  'db_name' => 'btc',
  'db_host' => 'localhost',
  'btc_protocol' => 'http',
  'btc_user' => 'root',
  'btc_password' => 'aaa',
  'btc_host' => 'localhost',
  'btc_port' => '18332',
  'text_logo' => 'PHPCoins',
  'site_name' => 'PHPCoins',
  'smtp_host' => 'smtp.yeah.net',
  'smtp_port' => '25',
  'smtp_user' => 'phpcoins@yeah.net',
  'smtp_password' => 'abcdef',
  'btc_decimal' => '8',
);
```

20:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\analytics.html

```
<div class="well well-sm">
  :<?=db("select * from trade where status=1 and ctime > {$GLOBALS['ONE_DAY_AGO']} order
by ctime desc")->fetch()['price']?>
  &nbsp;&nbsp;
  :<?=db("select * from trade where type=0 and status=1 and ctime >
{$GLOBALS['ONE_DAY_AGO']} order by ctime desc")->fetch()['price'] ?>
  &nbsp;&nbsp;
  :<?=db("select * from trade where type=1 and status=1 and ctime >
{$GLOBALS['ONE_DAY_AGO']} order by ctime desc")->fetch()['price']?>
  &nbsp;&nbsp;
  :<?=db("select * from trade where status=1 and ctime > {$GLOBALS['ONE_DAY_AGO']} order
by price desc")->fetch()['price']?>
  &nbsp;&nbsp;
  :<?=db("select * from trade where status=1 and ctime > {$GLOBALS['ONE_DAY_AGO']} order
by price asc")->fetch()['price']?>
</div>
```

21:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\asset.html

```
<ul class="list-group">
  <li class="list-group-item">:<?=user_info()['rmb']?></li>
  <li class="list-group-item">:<?=user_info()['rmb_frozen']?></li>
  <li class="list-group-item">:<?=user_info()['btc']?></li>
  <li class="list-group-item">:<?=user_info()['btc_frozen']?></li>
  <li class="list-group-item">
:<?=user_info()['rmb']+user_info()['rmb_frozen']+(user_info()['btc']+user_info()['btc_frozen'])*db("sel
ect price from trade where id=(select buy_trade  from deal order by ctime desc limit 1)")-
>fetch()['price']?></li>
</ul>
```

22:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\btc_record.html

```
<table>
  <thead>
```

```
    <th></th>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
  </thead>
<? foreach(db_page("select * from transfer where user_id=? order by ctime desc",[mid()]) as
$v):?>
  <tr>
    <td>
      <?=$v['address']?>
    </td>
    <td>
      <?=$v['amount']?>
    </td>
    <td>
      <?=date_full($v['ctime'])?>
    </td>
    <td>
      <?=$v['status'] == 0 ? '' : ''?>
    </td>
    <td>
      <a href="http://blockexplorer.com/address/<?=$v['address']?>" target="_blank"></a>
    </td>
  </tr>
<? endforeach?>
</table>
<?=$GLOBALS['pager']?>
```

```
23:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\buy_btc.html
<div well>
  <form act="buy_btc">
    <?=@form_text(['label'=>'','id'=>'price','col'=>[2,3]])?>
    <?=@form_text(['label'=>'','id'=>'amount'])?>
    <?=@form_password(['label'=>'','id'=>'pin'])?>
    <? if(user_info()['auth_trade']):?>
    <?=@form_text(['label'=>'','id'=>'auth'])?>
    <? endif?>
    <?=@form_submit('')?>
  </form>
</div>
```

24:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\change_deal_password.html

```
<div well>
  <form act="change_pin">
    <?=@form_password(['label'=>':','id'=>'old','col'=>[2,3]])?>
    <?=@form_password(['label'=>':','id'=>'pin'])?>
    <?=@form_password(['label'=>':','id'=>'confirm'])?>
    <?=@form_submit('')?>
  </form>
</div>
```

25:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\change_login_password.html

```
<div well>
  <form act="change_password">
    <?=@form_password(['label'=>':','id'=>'old','col'=>[2,3]])?>
    <?=@form_password(['label'=>':','id'=>'password'])?>
    <?=@form_password(['label'=>':','id'=>'confirm'])?>
    <?=@form_submit('')?>

    </div>
  </form>
</div>
```

26:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\change_profile.html

```
<div well>
  <form act="change_profile">
    <?=@form_text(['label'=>':','id'=>'name','value'=>user_info()['name'],'col'=>[1,3]])?>
    <?=@form_submit('')?>
  </form>
</div>
```

27:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\deal_chart.html

```
<div class="well">
  <div id="graphbox"></div>
</div>
<script>
 $(document).ready(function(){
   var datas =<?=json_encode(ohlc())?> , rates = [], vols = [];
   for(i = 0; i < datas.length; i++){
     rates.push([datas[i][0], parseFloat(datas[i][2],10), parseFloat(datas[i][3],10),
parseFloat(datas[i][4],10), parseFloat(datas[i][5],10)]);
     vols.push([datas[i][0], parseFloat(datas[i][1],10)]);
   }

   Highcharts.setOptions({
     lang: {
       loading: 'Loading...',
       months: ['', '', '', '', '', '', '', '', '', '', '', ''],
       shortMonths: ['', '', '', '', '', '', '', '', '', '', '', ''],
       weekdays: ['', '', '', '', '', '', ''],
       decimalPoint: '.',
       numericSymbols: ['k', 'M', 'G', 'T', 'P', 'E'],
       resetZoom: 'Reset zoom',
       resetZoomTitle: 'Reset zoom level 1:1',
       thousandsSep: ','
     },
     credits: {enabled: false},
     global:{useUTC:false},
   });

   new Highcharts.StockChart({
     chart: { renderTo: 'graphbox'},
     xAxis: { type: 'datetime' },
     legend: { enabled: false },
     tooltip: { xDateFormat: '%Y-%m-%d %H:%M %A', changeDecimals: 4 },
     scrollbar: {enabled: false},
     navigator: {enabled: false},
     rangeSelector: {
       buttons: [
 {type: 'minute', count: 60, text: '1h'},
 {type: 'minute', count: 120,text: '2h'},
 {type: 'minute', count: 360,text: '6h'},
 {type: 'minute', count: 720,text: '12h'},
 {type: 'day',    count: 1,  text: '1d'},
 {type: 'all', text: ''}
       ],
       selected: 5,
       inputEnabled: false
     },
     yAxis: [
       {  title: { text: ' [rmb]' }},
       { title: { text: ' [btc]' }, opposite: true }
     ],
     series: [
       { animation: false, name: ' [BTC]', type: 'column',  marker: { enabled: false }, yAxis: 1, data:
vols },
       { animation: false, name: ' [RMB]', type: 'candlestick', marker: { enabled: false }, data: rates }
     ]
   });
 });
</script>
```

28:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\email_validate.html

```
<div well>
  <?=user_info()['email']?> ,
 <form act="email_validate_send"><button class="btn btn-primary"></button></form>
</div>
```

29:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\enable_two_step_auth.html

```
<div well>
 <div class="container">
<? if(!user_info()['secret_installed']):?>
   <p>1. iPhoneApp StoreGoogle Authenticator<a href="http://itunes.apple.com/cn/app/google-
authenticator/id388497605?mt=8"></a></p>
<p>2. Android“”Google Authenticator<a href="http://apk.hiapk.com/html/2013/07/1643619.html">
</a>
</p>
   <p></p>
   <img src="<?=google_auth_qr_url()?>">
   <br><br>
<? endif?>
   <p></p>
   <form act="auth_config">
     <?=@form_checkbox(['label'=>'','id'=>'auth','col'=>[1,3],'item'=>[
     ['value'=>'trade','checked'=>user_info()['auth_trade'],'label'=>''],
     ['value'=>'withdraw','checked'=>user_info()['auth_withdraw'],'label'=>'']]])?>
     <?=@form_text(['label'=>':','id'=>'code'])?>
     <?=@form_submit('')?>
   </form>
 </div>
</div>
```

30:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\footer.html

```
<br>
<? if(DEV):?>
<div class="container">
```

```
  <div class="row">
    <div class="col-md-12">
      <div class="well">
  <? if(!empty($_SESSION['user_id'])):?>
<p>: UID:<?=user_info()['id']?> :<?=google_auth_code()?></p>
<? endif?>
:
<? foreach(db("select * from user") as $v):?>
<a href="<?=url('change_uid',['uid'=>$v['id']])?>"><?=$v['email']?></a>
<? endforeach?>
      </div>
    </div>
  </div>
</div>
<? endif?>
  </body>
</html>
```

31:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\header.html

```
<!DOCTYPE html>
<html>
  <head>
 <meta http-equiv="X-UA-Compatible" content="IE=edge">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta charset="utf-8">
    <title><?=@$GLOBALS['title']?:config('site_name')?></title>
    <link href="/public/css/bootstrap.min.css" rel="stylesheet" media="screen">
    <link href="/public/css/btc.css" rel="stylesheet" media="screen">
    <script>
     var ADMIN = <?=ADMIN?>
    </script>
  </head>
  <body>
    <script src="/public/js/jquery.js"></script>
    <script src="/public/js/bootstrap.js"></script>
    <script src="/public/js/highstock.src.js"></script>
    <script src="/public/js/btc.js"></script>
    <script src="/public/js/respond.js"></script>
```

32:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\help.html

```
<div class="container">
```

```
  <div class="row">
    <div class="col-md-4">
      <? html('help_left')?>
    </div>
    <div class="col-md-8">
      <? html('help_right')?>
    </div>
  </div>
</div>
```

33:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\help_left.html

```
<div class="list-group">
<? foreach(db("select * from help order by sort desc")->fetchAll() as $v):?>
  <a href="<?=url('help',['id'=>$v['id']])?>" class="list-group-item">
    <?=$v['title']?>
  </a>
<? endforeach?>
</div>
```

34:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\help_right.html

```
<? if(!empty($_REQUEST['id'])):?>
<div class="well">
  <p class="lead text-center">
    <?=help(['id'=>@$_REQUEST['id']])['title']?>
  </p>
  <p>
    <?=nl2br(help(['id'=>@$_REQUEST['id']])['body'])?>
  </p>
</div>
<? endif?>
```

35:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\index_body.html

```
<div class="container">
  <div class="row">
    <div class="col-md-9">
      <? html(['analytics','deal_chart','latest_order'])?>
    </div>
    <div class="col-md-3">
      <? html('right')?>
    </div>
  </div>
</div>
```

```
36:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\latest_deal.html
<div class="panel panel-default">
  <div class="panel-heading">
    <h4 class="panel-title">

    </h4>
  </div>
    <table class="table">
      <thead>
<th></th>
<th></th>
<th></th>
      </thead>
<? foreach(db("select * from trade where status=1 order by ctime desc limit 10")->fetchAll() as
$v):?>
      <tr>
<td>
  <?=$v['type']==0?'':''?>
</td>
<td>
  <?=$v['price']?>
</td>
<td>
  <?=$v['amount']?>
</td>
      </tr>
<? endforeach?>
    </table>

</div>

37:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\latest_order.html
<div class="row">
  <div class="col-md-6">
    <table>
      <thead>
<th></th>
<th></th>
<th></th>
      </thead>
<? $k=0; foreach(db("select * from trade where status=0 and type=0 order by price desc limit 10")-
>fetchAll() as $v):$k++;?>
      <tr>
<td>
  (<?=$k?>)
</td>
<td>
  <?=$v['amount']?>
</td>
<td>
  <?=$v['price']?>
</td>
      </tr>
<? endforeach?>
    </table>
  </div>
  <div class="col-md-6">
    <table>
      <thead>
<th></th>
<th></th>
<th></th>
      </thead>
<? $k=0; foreach(db("select * from trade where status=0 and type=1 order by price asc limit 10")-
>fetchAll() as $v):$k++;?>
      <tr>
<td>
  <?=$v['price']?>
</td>
<td>
  <?=$v['amount']?>
</td>
<td>
  (<?=$k?>)
</td>
      </tr>
<? endforeach?>

    </table>
  </div>
</div>
```

38:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\msg.html

```
<? html(['header','navbar']) ?>

<div class="container">
  <div class="row">
    <div class="col-md-12">
      <div class="well text-center">
<?=$_SESSION['msg']?>
      </div>
    </div>
  </div>
</div>
<? html('footer')?>
```

39:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\my_body.html

```
<div class="container">
  <div class="row">
    <div class="col-md-2">
      <? html('my_left')?>
    </div>
    <div class="col-md-10">
      <? html(@$_GET['show']?:'asset')?>
    </div>
  </div>
</div>
```

40:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\my_left.html

```
<div class="panel-group" id="my">
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">
<a data-toggle="collapse" data-parent="#my" href="#balance">

</a>
      </h4>
    </div>
    <div id="balance" class="panel-collapse collapse in">
      <div class="list-group no-margin-bottom">
<a href="<?=url('my',['show'=>'rmb_recharge'])?>" class="list-group-item"></a>
<a href="<?=url('my',['show'=>'rmb_withdraw'])?>" class="list-group-item"></a>
<a href="<?=url('my',['show'=>'transfer_btc_in'])?>" class="list-group-item"></a>
<a href="<?=url('my',['show'=>'transfer_btc_out'])?>" class="list-group-item"></a>
      </div>
    </div>
  </div>
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">
<a data-toggle="collapse" data-parent="#my" href="#order">

</a>
      </h4>
    </div>
    <div id="order" class="panel-collapse collapse in">
      <div class="list-group no-margin-bottom">
<a href="<?=url('my',['show'=>'buy_btc'])?>" class="list-group-item"></a>
<a href="<?=url('my',['show'=>'sell_btc'])?>" class="list-group-item"></a>
      </div>
    </div>
  </div>
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">
<a data-toggle="collapse" data-parent="#my" href="#account">

</a>
      </h4>
    </div>
    <div id="account" class="panel-collapse collapse in">
      <div class="list-group no-margin-bottom">
<a href="<?=url('my',['show'=>'trade_record'])?>" class="list-group-item"></a>
<a href="<?=url('my',['show'=>'rmb_recharge_record'])?>" class="list-group-item"></a>
<a href="<?=url('my',['show'=>'rmb_withdraw_record'])?>" class="list-group-item"></a>
<a href="<?=url('my',['show'=>'btc_record'])?>" class="list-group-item"></a>
      </div>
    </div>
  </div>
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">
<a data-toggle="collapse" data-parent="#my" href="#security">

</a>
      </h4>
    </div>
    <div id="security" class="panel-collapse collapse in">
      <a href="<?=url('my',['show'=>'email_validate'])?>" class="list-group-item"></a>
      <a href="<?=url('my',['show'=>'change_login_password'])?>" class="list-group-item"></a>
      <a href="<?=url('my',['show'=>'change_deal_password'])?>" class="list-group-item"></a>
      <a href="<?=url('my',['show'=>'enable_two_step_auth'])?>" class="list-group-item"></a>
    </div>
  </div>
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">
<a data-toggle="collapse" data-parent="#my" href="#profile">

</a>
      </h4>
    </div>
    <div id="profile" class="panel-collapse collapse in">
      <div class="list-group no-margin-bottom">
<a href="<?=url('my',['show'=>'change_profile'])?>" class="list-group-item"></a>
      </div>
    </div>
  </div>
</div>

41:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\navbar.html
<nav class="navbar navbar-default" role="navigation">
  <div class="container">
    <div class="row">
      <div class="col-md-12">
<div class="navbar-header">
  <a class="navbar-brand" href="/"><?=config('text_logo')?></a>
</div>
<ul class="nav navbar-nav navbar-left">
  <li><a href="<?=url('help')?>"></a></li>
</ul>
<ul class="nav navbar-nav navbar-right">
  <? if(mid() <= 0):?>
  <li><a href="<?=url('blank',['show'=>'sign_in'])?>"></a></li>
  <li><a href="<?=url('blank',['show'=>'sign_up'])?>"></a></li>
  <li><a href="<?=url('blank',['show'=>'password_reset_send'])?>"></a></li>
  <? else:?>
  <li><a href="<?=url('my')?>"><?=@user_info()['email']?></a></li>
  <li><a href="<?=url('log_out')?>"></a></li>
  <? endif?>
  <?if(DEV):?>
  <li><a href="<?=url('index',['admin'=>1])?>"></a></li>
  <? endif?>
</ul>
      </div>
    </div>
  </div>
</nav>
```

42:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\password_reset.html

```
<div class="container">
  <div class="row">
    <div class="col-md-4 col-md-offset-4">
      <div well>
<form act="password_reset">
  <?=@form_password(['label'=>':','id'=>'password','col'=>[4,8]])?>
  <?=@form_password(['label'=>':','id'=>'confirm_password'])?>
  <?=@form_password(['label'=>':','id'=>'pin'])?>
  <?=@form_password(['label'=>':','id'=>'confirm_pin'])?>
  <?=@form_hidden(['id'=>'email','value'=>$_REQUEST['email']])?>
  <?=@form_hidden(['id'=>'key','value'=>$_REQUEST['key']])?>
  <?=@form_submit('')?>
</form>
      </div>
    </div>
  </div>
</div>
```

43:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\password_reset_send.html

```
<div class="container">
  <div class="row">
    <div class="col-md-4 col-md-offset-4">
      <div class="well">
<form act="password_reset_send">
  <?=@form_text(['label'=>':','id'=>'email','col'=>[3,9]])?>
  <?=@form_submit('')?>
</form>
      </div>
    </div>
  </div>
</div>
```

```
44:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\right.html
<? html('latest_deal')?>

45:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\rmb_recharge.html
<div well>

</div>

46:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\rmb_recharge_record.html
<table>
  <thead>
    <th></th>
    <th></th>
  </thead>
  <?foreach(db_page("select * from recharge where user_id=? order by ctime desc",[mid()]) as
$v):?>
  <tr>
    <td>
      <?=$v['amount']?>
    </td>
    <td>
      <?=date_full($v['ctime'])?>
    </td>
  </tr>
  <?endforeach?>
</table>
<?=$GLOBALS['pager']?>

47:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\rmb_withdraw.html
<div class="well clearfix">
  <form act="withdraw">
    <?=@form_static(['label'=>':','value'=>user_info()['rmb'],'col'=>[2,3]])?>
    <?=@form_text(['label'=>':','id'=>'amount'])?>
    <?=@form_text(['label'=>':','id'=>'bank'])?>
    <?=@form_text(['label'=>':','id'=>'card'])?>
    <?=@form_text(['label'=>':','id'=>'name'])?>
    <? if(user_info()['auth_withdraw']):?>
    <?=@form_text(['label'=>':','id'=>'auth'])?>
    <? endif?>
    <?=@form_submit('')?>
  </form>
```

```
</div>

48:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\rmb_withdraw_record.html
<table>
  <thead>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
  </thead>
<?foreach(db_page("select * from withdraw where user_id=? order by ctime desc",[mid()]) as
$v):?>
  <tr>
    <td>
      <?=$v['bank']?>
    </td>
    <td>
      <?=$v['card']?>
    </td>
    <td>
      <?=$v['name']?>
    </td>
    <td>
      <?=$v['amount']?>
    </td>
    <td>
      <?=date_full($v['ctime'])?>
    </td>
    <td>
      <?=$v['status'] == 0 ? '' : ''?>
    </td>
  </tr>
<?endforeach?>
</table>
<?=$GLOBALS['pager']?>

49:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\sell_btc.html
<div class="well clearfix">
  <form act="sell_btc">
    <?=@form_text(['label'=>':','id'=>'price','col'=>[2,3]])?>
    <?=@form_static(['label'=>':','value'=>user_info()['btc']])?>
    <?=@form_text(['label'=>':','id'=>'amount'])?>
    <?=@form_password(['label'=>':','id'=>'pin'])?>
    <? if(user_info()['auth_trade']):?>
    <?=@form_text(['label'=>':','id'=>'auth'])?>
    <? endif?>
    <?=@form_submit('')?>
  </form>
</div>

50:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\sign_in.html
<div class="container">
  <div class="row">
    <div class="col-md-4 col-md-offset-4">
      <div class="well">
<form act="sign_in">
  <?=@form_text(['id'=>'email','label'=>':','col'=>[3,8]])?>
  <?=@form_password(['id'=>'password','label'=>':'])?>
  <?=@form_submit('')?>
</form>
      </div>
    </div>
  </div>
</div>

51:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\sign_up.html
<div class="container">
  <div class="row">
    <div class="col-md-6 col-md-offset-3">
      <div well>
<form act="sign_up">
  <?=@form_text(['label'=>':','id'=>'email','col'=>[4,5]])?>
  <?=@form_password(['label'=>':','id'=>'password'])?>
  <?=@form_password(['label'=>':','id'=>'confirm_password'])?>
  <?=@form_password(['label'=>':','id'=>'pin'])?>
  <?=@form_password(['label'=>':','id'=>'confirm_pin'])?>
  <?=@form_submit('')?>
</form>
      </div>
    </div>
  </div>
</div>
```

52:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\trade_record.html

```
<table>
  <thead>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
    <th></th>
  </thead>
<?foreach(db_page("select * from trade where user_id=? order by ctime desc",[mid()]) as $v):?>
 <tr>
   <td>
     <?=$v['type'] == 0 ? '' : ''?>
   </td>
   <td>
     <?=$v['price']?>
   </td>
   <td>
     <?=$v['amount']?>
   </td>
   <td>
     <?=$v['deal']?>
   </td>
   <td>
     <?=btcval($v['amount']-$v['deal'])?>
   </td>
   <td>
     <?=date('Ymd H:i:s',$v['ctime'])?>
   </td>
   <td>
     <?=$v['status'] == 0 ? '' : ($v['status'] == 1 ? '' : '')?>
   </td>
   <td>
     <?=$v['status'] == 0 ? '<a href="/?act=trade_close&id='.$v['id'].'"></a>':''?>
   </td>
 </tr>

<?endforeach?>
```

```
</table>
<?=$GLOBALS['pager']?>

53:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\transfer_btc_in.html
<div class="well">
  <p></p>
  <p class="text-center lead"><?=user_info()['address']?></p>
  <p></p>
  <p><a href="http://testnet.mojocoin.com/" target="_blank"></a>
</div>

54:F:\git\coin\\okbase.net\PHPCoins_V1.0\front\transfer_btc_out.html
<div well>
  <form act="transfer">
    <?=@form_static(['label'=>':','value'=>user_info()['btc'],'col'=>[2,3]])?>
    <?=@form_text(['label'=>':','id'=>'address'])?>
    <?=@form_text(['label'=>':','id'=>'amount'])?>
    <? if(user_info()['auth_withdraw']):?>
    <?=@form_text(['label'=>':','id'=>'auth'])?>
    <? endif?>
    <?=@form_submit('')?>
  </form>
</div>

55:F:\git\coin\\okbase.net\PHPCoins_V1.0\inc.php
<?php

function captcha($code=null){
  if( 0 == func_num_args()){
    require LIB.'/captcha/simple-php-captcha.php';
    $_SESSION['captcha'] = simple_php_captcha();
    return $_SESSION['captcha']['image_src'];
  }else{
    return strtolower($code) == strtolower($_SESSION['captcha']['code']);
  }
}

function remove_dupchar($str,$char,$time=2){
  return preg_replace_callback('#(['.preg_quote($char).']{'.$time.',})#',
       function($m){
 return $m[1]{1};
       }
       ,$str);
}

function multi_require($files,$ext='php'){
  $work_dir = empty(dirname(debug_backtrace()[0]['file'])) ? '' :
dirname(debug_backtrace()[0]['file']).'/';
  array_map(function($file)use($ext,$work_dir){
      require $work_dir.$file.($ext? '.'.$ext : '');
    },$files);
}

function html($file_name){
  foreach((array)$file_name as $f){
    include HTML.'/'.$f.'.html';
  }
}

function array_pick($keys,$array){
  foreach($keys as $k)
    $pick[$k] = $array[$k];
  return $pick;
}

function config($key='',$value=''){
  $config = read_array(ROOT.'/config.php');
  switch (func_num_args()){
  case 0:
    return $config;
    break;
  case 1:
    return @$config[$key];
    break;
  case 2:
    $config[$key] = $value;
    array_save(ROOT.'/config.php',$config);
    break;
  }
}

function array_save($file_name,$array=[]){
  $code = output_capture(function()use($array){
      var_export($array);
    });
    file_put_contents($file_name,<<<PHP
<?php
      return $code;
PHP
      );

}

function read_array($file_name){
  $array = include $file_name;
  $array === 1 && $array = [];
  return (array)$array;
}

function output_capture($output_code){
  ob_start();
  $output_code();
  $output = ob_get_contents();
  ob_end_clean();
  return $output;
}

function pdo(){
  static $db = null;
  !$db
    && ($db = new
PDO('mysql:dbname='.config()['db_name'].';host='.config()['db_host'],config()['db_user'],config()['db
_password']))
    && $db->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION)
    && $db->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE,PDO::FETCH_ASSOC);
  return $db;
}

function error($msg=''){
  $msg = 'File: '.debug_backtrace()[0]['file'].' Line: '.debug_backtrace()[0]['line'];
  echo is_ajax() ? json_encode(['status'=>0,'msg'=>$msg]) : $msg;
}

function db($prepare='',$params=[],$bool=0){
  if(!is_array($params)){
    $params = [$params];
  }
  if(!$prepare)
    return pdo();
  $q = db()->prepare($prepare);
  $status = $q->execute($params);
  if($bool == 0){
    return $q;
  }elseif($bool == 1){
    return $status;
  }elseif($bool == 2){
    return pdo();
  }
}

function is_ajax(){
  return isset($_SERVER['HTTP_X_REQUESTED_WITH']) and
strtolower($_SERVER['HTTP_X_REQUESTED_WITH']) == 'xmlhttprequest';
}

function db_clear(){
  array_map(function($table_name){
db("delete from $table_name");
    },array_column(db("show tables")->fetchAll(),'Tables_in_'.config('db_name')));
}

function times($time,$func){
  for($i=0;$i<$time;$i++){
    $func();
  }
}

function bool2int($data){
  if($data === true){
    return 1;
  }elseif($data === false){
    return 0;
  }elseif(is_array($data)){
    foreach($data as $k=>$v){
      if($v === true){
$data[$k] = 1;
      }elseif($v === false){
$data[$k] = 0;
```

```
      }elseif(is_array($v)){
$data[$k] = bool2int($v);
      }
    }
  }
  return $data;
}

function call_user_func_hash($function_name,$hash){
  $arg = [];
  foreach(is_array($function_name)
  ? (new ReflectionClass($function_name[0]))->getMethod($function_name[1])->getParameters()
  :(new ReflectionFunction($function_name))->getParameters() as $param){
    $arg[] = @$hash[$param->name];
  }
  return call_user_func_array($function_name,$arg);
}

function google_auth(){
  require_once LIB.'/auth/PHPGangsta/GoogleAuthenticator.php';
  return new PHPGangsta_GoogleAuthenticator();
}

function google_auth_create_secret(){
  return google_auth()->createSecret();
}

function google_auth_qr_url(){
  return preg_replace('#^https#','http',google_auth()-
>getQRCodeGoogleUrl(config('text_logo'),google_auth_current_user_secret()));
}

function google_auth_code(){
  return google_auth()->getCode(google_auth_current_user_secret());
}

function google_auth_verify($code){
  return google_auth()->verifyCode(google_auth_current_user_secret(),$code,2);
}

function google_auth_current_user_secret(){
  return db("select * from user where id=?",[$_SESSION['user_id']])->fetch()['secret'];
```

```
}

function btc(){
  static $btc = null;
  require_once LIB.'/btc/bitcoin.inc';
  $btc = $btc ?: new
BitcoinClient(config('btc_protocol'),config('btc_user'),config('btc_password'),config('btc_host'),confi
g('btc_port'),'',0);
  return $btc;
}

function user_info($id=null){
  if(!$id && empty($_SESSION['user_id']))
    return;
  $id = $id ?: $_SESSION['user_id'];
  return array_hash_only(db("select * from user where id=?",[$id])->fetch());
}

function array_hash_only($array){
  foreach($array as $k=>$v){
    if(is_numeric($k))
      unset($array[$k]);
  }
  return $array;
}

function receive_btc(){
  db()->exec('lock tables user write');
  if($received = btc()->getreceivedbyaddress(user_info()['address'],0) - user_info()['received']){
    db("update user set btc=btc+?,received=received+? where
id=?",[$received,$received,$_SESSION['user_id']],1);
  }
  db()->exec('unlock tables');
}



function db_lock($tables){
  $tables = is_array($tables) ? $tables : [$tables];
  $tables = join(',', array_map(function($t){
      return $t.' write';
      },$tables));
  db()->exec('set autocommit=0');
  db()->exec('lock tables '.$tables);
}

function db_unlock(){
  db()->exec('commit');
  db()->exec('unlock tables');
}

function db_rollback(){
  db()->exec('rollback');
}

function pager($count,$per_page=20){
  $cur_page = empty($_GET['p']) ? 1 : $_GET['p'];
  $total_page = ceil($count/$per_page);
  if($total_page <= 1)
    return '';
  $page_url = function($p){
    $url = parse_url($_SERVER['REQUEST_URI']);
    $query=[];
    parse_str(@$url['query'],$query);
    $query = array_merge($query,['p'=>$p]);
    return $url['path'].'?'.http_build_query($query);
  };
  $pre_url = $cur_page > 1 ? $page_url($cur_page-1) : '';
  $next_url = $cur_page < $total_page ? $page_url($cur_page+1) : '';
  $ul_start = '<ul class="pager">';
  $ul_end = '</ul>';
  $html = $ul_start
    .($pre_url ? '<li><a href="'.$pre_url.'"></a></li>' : '<li class="disabled"><a href="#"></a></li>')
    .($next_url ? '<li><a href="'.$next_url.'"></a></li>' : '<li class="disabled"><a href="#"></a></li>')
    .$ul_end;
  return $html;
}

function run_deal(){
  db_lock(['user','deal','trade']);
  for(;;){
    $buy = db("select * from trade where type=0 and status=0 order by price desc")->fetch();
    $sell = db("select * from trade where type=1 and status=0 order by price asc")->fetch();
    if($buy && $sell && floatval($buy['price']) - floatval($sell['price']) >= 0){
      $amount =  min($buy['amount']-$buy['deal'],$sell['amount']-$sell['deal']);
      if(!(db("update trade set deal=deal+? where id in (?,?)",[$amount,$buy['id'],$sell['id']],1) &&
    db("update user set rmb_frozen=rmb_frozen-?,btc=btc+? where
id=?",[$amount*$buy['price'],$amount,$buy['user_id']],1) &&
    db("update user set btc_frozen=btc_frozen-?,rmb=rmb+? where
id=?",[$amount,$amount*$sell['price'],$sell['user_id']],1) &&
    db("insert into
deal(buy_trade,sell_trade,ctime,amount)values(?,?,?,?)",[$buy['id'],$sell['id'],time(),$amount],1) &&
    db("update trade set status=1 where amount=deal",[],1))){
    db_rollback();
      }
    }else{
      break;
    }
  }
  db_unlock();
}

function btcval($number){
  return floatval(number_format(abs($number),config('btc_decimal'),'.',''));
}

function _btcval(&$number){
  $number = btcval($number);
  return $number;
}

function rmbval($number){
  return floatval(number_format(abs($number),2,'.',''));
}

function _rmbval(&$number){
  $number = rmbval($number);
  return $number;
}

function ohlc(){
  $ohlc = [];
  $o = db("select price,floor(deal.ctime/60) min from deal inner join trade on
deal.buy_trade=trade.id inner join (select min(id) id from deal group by floor(ctime/60)) b on
deal.id=b.id")->fetchAll();
  $h = db("select max(price) price,floor(deal.ctime/60) min from deal inner join trade on
deal.buy_trade=trade.id group by floor(deal.ctime/60)")->fetchAll();
```

```
  $l = db("select min(price) price,floor(deal.ctime/60) min from deal inner join trade on
deal.buy_trade=trade.id  group by floor(deal.ctime/60)")->fetchAll();
  $c = db("select price,floor(deal.ctime/60) min from deal inner join trade on
deal.buy_trade=trade.id  inner join (select max(id) id from deal group by floor(ctime/60)) b on
deal.id=b.id")->fetchAll();
  $d = db("select sum(amount) amount,ctime,floor(ctime/60) min from deal group by
floor(ctime/60)")->fetchAll();

  $o = array_pk($o,'min');
  $h = array_pk($h,'min');
  $l = array_pk($l,'min');
  $c = array_pk($c,'min');
  $d = array_pk($d,'min');
  foreach($o as $v){
    $ohlc[] = [$d[$v['min']]['ctime']*1000,
       $d[$v['min']]['amount'],
       $o[$v['min']]['price'],
       $h[$v['min']]['price'],
       $l[$v['min']]['price'],
       $c[$v['min']]['price']];
  }
  return $ohlc;
}

function array_pk($array,$pk){
  $n_array = [];
  foreach($array as $v){
    $n_array[$v[$pk]] = $v;
  }
  return $n_array;
}

function send_mail($email,$title,$body){
  require_once LIB.'/mail/PHPMailerAutoload.php';
  $mail = new PHPMailer();
  $mail->isSMTP();
  $mail->SMTPDebug = 0;
  $mail->Debugoutput = 'html';
  $mail->Host = config('smtp_host');
  $mail->Port = config('smtp_port');
  $mail->SMTPSecure = 'tls';
  $mail->SMTPAuth = true;
```

```php
  $mail->Username = config('smtp_user');
  $mail->Password = config('smtp_password');
  $mail->setFrom(config('smtp_user'),config('text_logo'));
  $mail->addAddress($email,$email);
  $mail->Subject = $title;
  $mail->msgHTML($body);
  return $mail->send();
}

function array_remove_keys($array,$keys){
  $keys = is_array($keys) ? $keys : [$keys];
  foreach($array as $k=>$v){
    if(in_array($k,$keys))
      unset($array[$k]);
  }
  return $array;
}

function array_remove_number_keys($array){
  foreach($array as $k=>$v){
    if(preg_match('#^[0-9]+$#',$k))
      unset($array[$k]);
  }
  return $array;
}

function array_trim($array){
  foreach($array as $k=>$v){
    $array[$k] = is_array($v) ? array_trim($v) : trim($v);
  }
  return $array;
}

function get_external_ip(){
  return file_get_contents('http://ipecho.net/plain');
}

function date_full($timestamp){
  return date('Ymd H:i:s',$timestamp);
}

function array_map_suf($arr,$suf){
  return array_map(function($v)use($suf){
      return $v.$suf;
    },$arr);
}

function array_map_pre($arr,$pre){
  return array_map(function($v)use($pre){
      return $pre.$v;
    },$arr);
}

function db_curd(){
  array_map(function($t){
      $read = <<<PHP
   function $t(\$arr){
if(!is_array(\$arr)){
  \$arr = [table_pk('$t')=>\$arr];
}
return db("select * from $t where ".implode(' and
',array_map_suf(array_keys(\$arr),'=?')),array_values(\$arr))->fetch();
      }
PHP
  ;
    eval($read);
    $add = <<<PHP
      function {$t}_add(\$arr){
      if(db("insert into $t
(".implode(',',array_keys(\$arr)).")values(".implode(',',array_fill(0,count(\$arr),'?')).")",array_values(\
$arr),1)){
      return db()->lastInsertId();
      }
    }
PHP
      ;
    eval($add);
    $edit = <<<PHP
      function {$t}_edit(\$arr){
      \$sql = "update $t set ".
rtrim(
      implode(
      '',array_map_suf(
       array_map_pre(
     array_keys(
array_remove_keys(
  \$arr,table_pk('$t')
  )
)
     ,' '),'=?,'
       )
      ),','
      )
." where ".table_pk('$t')."=?";
      return db(\$sql,
array_merge(
    array_values(
 array_remove_keys(
   \$arr,table_pk('$t'))),
    [\$arr[table_pk('$t')]]),1);
    }
PHP
  ;
eval($edit);
$del = <<<PHP
  function {$t}_del(\$pk){
  return db("delete from $t where ".
    table_pk('$t').
    '=?',[\$pk],1);
}
PHP
  ;
eval($del);
    },array_column(db("show tables")->fetchAll(),'Tables_in_'.config('db_name')));
}

function table_pk($table_name){
  foreach(db("show columns from $table_name")->fetchAll() as $v){
    if($v['Key'] == 'PRI')
      return $v['Field'];
  }
}

function mid(){
  return intval(@$_SESSION['user_id']);
}
```

```
function msg($msg=''){
  $_SESSION['msg'] = $msg;
  jump(url('msg'));
}

function jump($url=''){
  $url = $url ?: $_SERVER['HTTP_REFERER'];
  header('location:'.$url);
  exit;
}

function db_page($sql,$arg=[],$per=20){
  $GLOBALS['pager'] = pager(db(preg_replace('#select .*? from #','select count(*) as count from
',$sql),$arg)->fetch()['count'],$per);
  return db($sql.' limit '.(intval(@$_REQUEST['p']?:1) - 1).','.$per,$arg)->fetchAll();
}

function url($act,$arg=[]){
  if(is_array($act)){
    $arg = $act;
    $act = ACT;
  }
  ADMIN && ($arg['admin'] = 1);
  return $_SERVER['REQUEST_SCHEME'].'://'.$_SERVER['HTTP_HOST'].'/?'.
http_build_query(array_merge(['act'=>$act],$arg));
}

function form_text($arg){
  $arg['col'] = $GLOBALS['COL'] =  $arg['col']?:$GLOBALS['COL'];
    return <<<HTML
<div class="form-group">
<label class="col-md-{$arg['col'][0]} control-label" for="{$arg['id']}">
      {$arg['label']}
</label>
<div class="col-md-{$arg['col'][1]}">
<input type="text" class="form-control" id="{$arg['id']}" name="{$arg['id']}" value="{$arg['value']}" />
</div>
</div>
HTML
    ;
}
```

```
function form_submit($arg){
  if(!is_array($arg))
    $arg = ['text'=>$arg];
  $arg['col'] = $GLOBALS['COL'] =  $arg['col']?:$GLOBALS['COL'];
  $arg['text'] = $arg['text']?:'Submit';
  return <<<HTML
<div class="form-group">
<div class="col-md-{$arg['col'][1]} col-md-offset-{$arg['col'][0]}">
<button class="btn btn-primary">
    {$arg['text']}
</button>
</div>
</div>
HTML
    ;
}

function form_static($arg){
  $arg['col'] = $GLOBALS['COL'] =  $arg['col']?:$GLOBALS['COL'];
  return <<<HTML
<div class="form-group">
<label class="control-label col-md-{$arg['col'][0]}" for="{$arg['id']}">
    {$arg['label']}
</label>
<div class="col-md-{$arg['col'][1]}">
<p class="form-control-static" id="{$arg['id']}">
    {$arg['value']}
</p>
</div>
</div>
HTML
    ;
}

function form_password($arg){
  $arg['col'] = $GLOBALS['COL'] =  $arg['col']?:$GLOBALS['COL'];
  return <<<HTML
<div class="form-group">
<label class="col-md-{$arg['col'][0]} control-label" for="{$arg['id']}">
    {$arg['label']}
</label>
```

```
<div class="col-md-{$arg['col'][1]}">
<input type="password" class="form-control" id="{$arg['id']}" value="{$arg['value']}"
name="{$arg['id']}" />
</div>
</div>
HTML
    ;
}

function form_checkbox($arg){
  $arg['col'] = $GLOBALS['COL'] =  $arg['col']?:$GLOBALS['COL'];
  $html =  <<<HTML
<div class="form-group">
<label class="col-md-{$arg['col'][0]} control-label">
    {$arg['label']}
</label>
HTML
    ;
  foreach($arg['item'] as $v){
    $checked = $v['checked'] ? 'checked' : '';
    $html .= <<<HTML
<label class="control-label">
<input type="checkbox" name="{$arg['id']}[]" value="{$v['value']}" $checked />
      {$v['label']}
</label>
HTML
      ;
  }
  $html .= <<<HTML
</div>
HTML
    ;
  return $html;
}

function form_textarea($arg){
  $arg['col'] = $GLOBALS['COL'] =  $arg['col']?:$GLOBALS['COL'];
  $arg['rows'] = $arg['rows'] ?: 3;
  return <<<HTML
<div class="form-group">
<label class="control-label col-md-{$arg['col'][0]}" for="{$arg['id']}">
    {$arg['label']}
```

```
</label>
<div class="col-md-{$arg['col'][1]}">
    <textarea class="form-control" id="{$arg['id']}" name="{$arg['id']}"
rows="{$arg['rows']}">{$arg['value']}</textarea>
</div>
</div>
HTML
    ;
}

function form_hidden($arg){
  $arg['col'] = $GLOBALS['COL'] =  $arg['col']?:$GLOBALS['COL'];
  return <<<HTML
<input type="hidden" name="{$arg['id']}" value="{$arg['value']}" id="{$arg['id']}" />
HTML
    ;
}

function form_captcha($arg=null){
  $arg['col'] = $GLOBALS['COL'] =  $arg['col']?:$GLOBALS['COL'];
  $arg['captcha'] = $arg['captcha']?:captcha();
  return <<<HTML
<div class="form-group">
<div class="col-md-{$arg['col'][1]} col-md-offset-{$arg['col'][0]}">
<img src="{$arg['captcha']}">
</div>
</div>
HTML
    ;
}

function output($content){
  echo is_ajax() ? json_encode(bool2int($content)) : $content;
}

function v_email($email){
  return 1 == preg_match('/[a-z0-9]+@[a-z0-9]+\.[a-z0-9]+/',$email);
}

function v_assert($validators){
  $validators = is_array($validators) ? $validators : [$validators];
  foreach($validators as $v){
    if(is_callable($v)){
      if(($msg = $v) !== true){
v_msg($msg);
      }
    }else{
      if($v !== true){
v_msg($v);
      }
    }
  }
}

function v_msg($msg){
  output(['msg'=>$msg]);
  exit;
}

function _trim(&$str,$char=" \t\n\r\0\x0B"){
  $str = trim($str,$char);
  return $str;
}

function _intval(&$number){
  $number = intval($number);
  return $number;
}

56:F:\git\coin\\okbase.net\PHPCoins_V1.0\index.php
<?php
define('DEV',0);
DEV || error_reporting(0);
@session_start();
define('ROOT',dirname(__FILE__));
define('LIB',dirname(__FILE__).'/lib');
define('ADMIN',empty($_REQUEST['admin']) ? 0 : 1);
define('HTML',ROOT.'/'.(ADMIN ? 'back' : 'front'));
define('ACT',empty($_REQUEST['act']) ? 'index' : $_REQUEST['act']);
require_once ROOT.'/inc.php';
require_once ROOT.'/act.php';
require_once ROOT.'/admin.php';
require_once ROOT.'/init.php';
```

57:F:\git\coin\\okbase.net\PHPCoins_V1.0\init.php

```php
<?php
$ONE_DAY_AGO = time() - 3600*24;
require_once LIB.'/fire/fb.php';

date_default_timezone_set('Asia/Shanghai');
db_curd();
```

```
if(!file_exists(ROOT.'/install.lock')){
  require ROOT.'/install.php';
}
$_REQUEST = array_trim($_REQUEST);
if(ADMIN && empty($_SESSION['admin']) && !in_array(ACT,['sign_in','do_sign_in'])){
  jump('/?act=sign_in&admin=1');
}
if(!ADMIN && empty(mid()) &&
!in_array(ACT,['blank','help','index','sign_in','sign_up','password_reset_send','password_reset'])){
  jump('/?act=blank&show=sign_in');
}
$act = ADMIN ? (new admin) : (new act);
if( method_exists($act,ACT)){
  output(call_user_func_hash([$act,ACT],$_REQUEST));
}else{
  html(ACT);
}
```

58:F:\git\coin\\okbase.net\PHPCoins_V1.0\install.php

```
<?php

version_compare(PHP_VERSION,'5.5.0','>=') || die('php5.5.0');
class_exists('PDO') || die('pdo');
extension_loaded('gd') && function_exists('gd_info') || die('gd');
array_map(function($sql){
    if(!db(trim($sql),[],1)){
      die('db error');
    }
  },explode(';',file_get_contents(ROOT.'/install.sql')));
file_put_contents(ROOT.'/install.lock','');
```

59:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\auth\example\example1.php

```
<?php

require_once '../PHPGangsta/GoogleAuthenticator.php';

$ga = new PHPGangsta_GoogleAuthenticator();
```

```
//$secret = $ga->createSecret();
$secret = file_get_contents('secret');
echo "Secret is: ".$secret."\n\n<br>";
file_put_contents('secret',$secret);
$qrCodeUrl = $ga->getQRCodeGoogleUrl('BTC', $secret);
echo "Google Charts URL for the QR-Code: ".$qrCodeUrl."\n\n<br>";
echo <<<HTML
<img src="$qrCodeUrl">
HTML
;



$oneCode = $ga->getCode($secret);
echo "Checking Code '$oneCode' and Secret '$secret':\n<br>";

$checkResult = $ga->verifyCode($secret, $oneCode, 2);    // 2 = 2*30sec clock tolerance
if ($checkResult) {
    echo 'OK';
} else {
    echo 'FAILED';
}
```

60:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\auth\PHPGangsta\GoogleAuthenticator.php

```
<?php

/**
 * PHP Class for handling Google Authenticator 2-factor authentication
 *
 * @author Michael Kliewe
 * @copyright 2012 Michael Kliewe
 * @license http://www.opensource.org/licenses/bsd-license.php BSD License
 * @link http://www.phpgangsta.de/
 */

class PHPGangsta_GoogleAuthenticator
{
    protected $_codeLength = 6;

    /**
     * Create new secret.
     * 16 characters, randomly chosen from the allowed base32 characters.
     *
     * @param int $secretLength
     * @return string
     */
    public function createSecret($secretLength = 16)
    {
        $validChars = $this->_getBase32LookupTable();
        unset($validChars[32]);

        $secret = '';
        for ($i = 0; $i < $secretLength; $i++) {
            $secret .= $validChars[array_rand($validChars)];
        }
        return $secret;
    }

    /**
     * Calculate the code, with given secret and point in time
     *
     * @param string $secret
     * @param int|null $timeSlice
     * @return string
     */
    public function getCode($secret, $timeSlice = null)
    {
        if ($timeSlice === null) {
            $timeSlice = floor(time() / 30);
        }

        $secretkey = $this->_base32Decode($secret);

        // Pack time into binary string
        $time = chr(0).chr(0).chr(0).chr(0).pack('N*', $timeSlice);
        // Hash it with users secret key
        $hm = hash_hmac('SHA1', $time, $secretkey, true);
        // Use last nipple of result as index/offset
        $offset = ord(substr($hm, -1)) & 0x0F;
        // grab 4 bytes of the result
        $hashpart = substr($hm, $offset, 4);

        // Unpak binary value
        $value = unpack('N', $hashpart);
```

```
        $value = $value[1];
        // Only 32 bits
        $value = $value & 0x7FFFFFFF;

        $modulo = pow(10, $this->_codeLength);
        return str_pad($value % $modulo, $this->_codeLength, '0', STR_PAD_LEFT);
    }

    /**
     * Get QR-Code URL for image, from google charts
     *
     * @param string $name
     * @param string $secret
     * @return string
     */
    public function getQRCodeGoogleUrl($name, $secret) {
        $urlencoded = urlencode('otpauth://totp/'.$name.'?secret='.$secret.'');
        return
'https://chart.googleapis.com/chart?chs=200x200&chld=M|0&cht=qr&chl='.$urlencoded.'';
    }

    /**
     * Check if the code is correct. This will accept codes starting from $discrepancy*30sec ago to
$discrepancy*30sec from now
     *
     * @param string $secret
     * @param string $code
     * @param int $discrepancy This is the allowed time drift in 30 second units (8 means 4 minutes
before or after)
     * @return bool
     */
    public function verifyCode($secret, $code, $discrepancy = 1)
    {
        $currentTimeSlice = floor(time() / 30);

        for ($i = -$discrepancy; $i <= $discrepancy; $i++) {
            $calculatedCode = $this->getCode($secret, $currentTimeSlice + $i);
            if ($calculatedCode == $code ) {
                return true;
            }
        }
        return false;
    }

    /**
     * Set the code length, should be >=6
     *
     * @param int $length
     * @return PHPGangsta_GoogleAuthenticator
     */
    public function setCodeLength($length)
    {
        $this->_codeLength = $length;
        return $this;
    }

    /**
     * Helper class to decode base32
     *
     * @param $secret
     * @return bool|string
     */
    protected function _base32Decode($secret)
    {
        if (empty($secret)) return '';

        $base32chars = $this->_getBase32LookupTable();
        $base32charsFlipped = array_flip($base32chars);

        $paddingCharCount = substr_count($secret, $base32chars[32]);
        $allowedValues = array(6, 4, 3, 1, 0);
        if (!in_array($paddingCharCount, $allowedValues)) return false;
        for ($i = 0; $i < 4; $i++){
            if ($paddingCharCount == $allowedValues[$i] &&
                substr($secret, -($allowedValues[$i])) != str_repeat($base32chars[32],
$allowedValues[$i])) return false;
        }
        $secret = str_replace('=','', $secret);
        $secret = str_split($secret);
        $binaryString = "";
        for ($i = 0; $i < count($secret); $i = $i+8) {
            $x = "";
            if (!in_array($secret[$i], $base32chars)) return false;
            for ($j = 0; $j < 8; $j++) {
                $x .= str_pad(base_convert(@$base32charsFlipped[@$secret[$i + $j]], 10, 2), 5, '0',
STR_PAD_LEFT);
            }
            $eightBits = str_split($x, 8);
            for ($z = 0; $z < count($eightBits); $z++) {
                $binaryString .= ( ($y = chr(base_convert($eightBits[$z], 2, 10))) || ord($y) == 48 ) ?
$y:"";
            }
        }
        return $binaryString;
    }

    /**
     * Helper class to encode base32
     *
     * @param string $secret
     * @param bool $padding
     * @return string
     */
    protected function _base32Encode($secret, $padding = true)
    {
        if (empty($secret)) return '';

        $base32chars = $this->_getBase32LookupTable();

        $secret = str_split($secret);
        $binaryString = "";
        for ($i = 0; $i < count($secret); $i++) {
            $binaryString .= str_pad(base_convert(ord($secret[$i]), 10, 2), 8, '0', STR_PAD_LEFT);
        }
        $fiveBitBinaryArray = str_split($binaryString, 5);
        $base32 = "";
        $i = 0;
        while ($i < count($fiveBitBinaryArray)) {
            $base32 .= $base32chars[base_convert(str_pad($fiveBitBinaryArray[$i], 5, '0'), 2, 10)];
            $i++;
        }
        if ($padding && ($x = strlen($binaryString) % 40) != 0) {
            if ($x == 8) $base32 .= str_repeat($base32chars[32], 6);
            elseif ($x == 16) $base32 .= str_repeat($base32chars[32], 4);
            elseif ($x == 24) $base32 .= str_repeat($base32chars[32], 3);
            elseif ($x == 32) $base32 .= $base32chars[32];
        }
        return $base32;
    }

    /**
     * Get array with all 32 characters for decoding from/encoding to base32
     *
     * @return array
     */
    protected function _getBase32LookupTable()
    {
        return array(
            'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', //  7
            'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', // 15
            'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', // 23
            'Y', 'Z', '2', '3', '4', '5', '6', '7', // 31
            '='  // padding char
        );
    }
}
```

61:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\captcha\index.php

```
<?php
session_start();
$_SESSION = array();

include("simple-php-captcha.php");
$_SESSION['captcha'] = simple_php_captcha();

?>
<!DOCTYPE html>
<html>
<head>
    <title>Example &raquo; A simple PHP CAPTCHA script</title>
    <style type="text/css">
    pre {
    border: solid 1px #bbb;
    padding: 10px;
    margin: 2em;
    }
    img {
    border: solid 1px #ccc;
    margin: 0 2em;
    }
    </style>
</head>
<body>
    <h1>
        CAPTCHA Example
    </h1>

    <h2>Usage</h2>

    <p>
    The following code will prepare a CAPTCHA image and keep the code in a session
    variable for later use:
    </p>

<pre>
&lt;?php
session_start();
include("simple-php-captcha.php");
$_SESSION['captcha'] = simple_php_captcha();
?&gt;
</pre>

    <p>
    After the call to <code>simple_php_captcha()</code> above,
    <code>$_SESSION['captcha']</code> will be something like this:
    </p>

<pre>
<?php
print_r($_SESSION['captcha']);
?>
</pre>

    <p>
    To display the CAPTCHA image, create an HTML <code>&lt;img&gt;</code> using
    <code>$_SESSION['captcha']['image_src']</code> as the <code>src</code> attribute:
    </p>
    <p>
    <?php
    echo '<img src="' . $_SESSION['captcha']['image_src'] . '" alt="CAPTCHA code">';

    ?>
    </p>

    <p>
    To verify the CAPTCHA value on the next page load (or in an AJAX request), test
    against  <code>$_SESSION['captcha']['code']</code>. You can use
    <code>strtolower()</code> or <code>strtoupper()</code> to perform a
    case-insensitive match.
    </p>

    <h2>Configuration</h2>
    <p>
    Configuration is easy and all values are optional. To specify one or more options,
    do this:
    </p>

<pre>
&lt;?php

$_SESSION['captcha'] = simple_php_captcha( array(
'min_length' => 5,
'max_length' => 5,
'backgrounds' => array(image.png', ...),
'fonts' => array('font.ttf', ...),
'characters' => 'ABCDEFGHJKLMNPRSTUVWXYZabcdefghjkmnprstuvwxyz23456789',
'min_font_size' => 28,
'max_font_size' => 28,
'color' => '#666',
'angle_min' => 0,
'angle_max' => 10,
'shadow' => true,
'shadow_color' => '#fff',
'shadow_offset_x' => -1,
'shadow_offset_y' => 1
));

&gt;
</pre>
```

```
    <h2>Notes</h2>
    <ul>
    <li>
    <strong>Important!</strong> Make sure you call <code>session_start()</code> before
    calling the <code>simple_php_captcha()</code> function
    </li>
    <li>
    Requires PHP GD2 library
    </li>
    <li>
    Backgound images must be in PNG format
    </li>
    <li>
    Fonts must be either TTF or OTF
    </li>
    <li>
    Backgrounds and fonts must be specified using their full paths (tip: use
    <code>$_SERVER['DOCUMENT_ROOT'] . '/' . [path-to-file]</code>)
    </li>
    <li>
    Angles should not exceed approximately 15 degrees, as the text will sometimes
    appear outside of the viewable area
    </li>
    <li>
    Creates a function called <code>simple_php_captcha()</code> in the global namespace
    </li>
    <li>
    Uses the <code>$_SESSION['simple-php-captcha']</code> session variable
    </li>
    </ul>

</body>
</html>
```

62:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\captcha\simple-php-captcha.php

```
<?php
//
//A simple PHP CAPTCHA script
//
//Copyright 2013 by Cory LaViska for A Beautiful Site, LLC.
//
```

```
//See readme.md for usage, demo, and licensing info
//
function simple_php_captcha($config = array()) {

// Check for GD library
if( !function_exists('gd_info') ) {
throw new Exception('Required GD library is missing');
}

$bg_path = dirname(__FILE__) . '/backgrounds/';
$font_path = dirname(__FILE__) . '/fonts/';

// Default values
$captcha_config = array(
'code' => '',
'min_length' => 5,
'max_length' => 5,
'backgrounds' => array(
$bg_path . '45-degree-fabric.png',
$bg_path . 'cloth-alike.png',
$bg_path . 'grey-sandbag.png',
$bg_path . 'kinda-jean.png',
$bg_path . 'polyester-lite.png',
$bg_path . 'stitched-wool.png',
$bg_path . 'white-carbon.png',
$bg_path . 'white-wave.png'
),
'fonts' => array(
$font_path . 'times_new_yorker.ttf'
),
'characters' => 'ABCDEFGHJKLMNPRSTUVWXYZabcdefghjkmnprstuvwxyz23456789',
'min_font_size' => 28,
'max_font_size' => 28,
'color' => '#666',
'angle_min' => 0,
'angle_max' => 10,
'shadow' => true,
'shadow_color' => '#fff',
'shadow_offset_x' => -1,
'shadow_offset_y' => 1
);
// Overwrite defaults with custom config values
if( is_array($config) ) {
foreach( $config as $key => $value ) $captcha_config[$key] = $value;
}

// Restrict certain values
if( $captcha_config['min_length'] < 1 ) $captcha_config['min_length'] = 1;
if( $captcha_config['angle_min'] < 0 ) $captcha_config['angle_min'] = 0;
if( $captcha_config['angle_max'] > 10 ) $captcha_config['angle_max'] = 10;
if( $captcha_config['angle_max'] < $captcha_config['angle_min'] ) $captcha_config['angle_max'] =
$captcha_config['angle_min'];
if( $captcha_config['min_font_size'] < 10 ) $captcha_config['min_font_size'] = 10;
if( $captcha_config['max_font_size'] < $captcha_config['min_font_size'] )
$captcha_config['max_font_size'] = $captcha_config['min_font_size'];

// Use milliseconds instead of seconds
srand(microtime() * 100);

// Generate CAPTCHA code if not set by user
if( empty($captcha_config['code']) ) {
$captcha_config['code'] = '';
$length = rand($captcha_config['min_length'], $captcha_config['max_length']);
while( strlen($captcha_config['code']) < $length ) {
$captcha_config['code'] .= substr($captcha_config['characters'], rand() %
(strlen($captcha_config['characters'])), 1);
}
}

// Generate HTML for image src
$image_src = substr(__FILE__, strlen($_SERVER['DOCUMENT_ROOT'])) .
'?_CAPTCHA&amp;t=' . urlencode(microtime());
$image_src = '/' . ltrim(preg_replace('/\\\\/', '/', $image_src), '/');

$_SESSION['_CAPTCHA']['config'] = serialize($captcha_config);

return array(
'code' => $captcha_config['code'],
'image_src' => $image_src
);

}
```

```
if( !function_exists('hex2rgb') ) {
function hex2rgb($hex_str, $return_string = false, $separator = ',') {
$hex_str = preg_replace("/[^0-9A-Fa-f]/", '', $hex_str); // Gets a proper hex string
$rgb_array = array();
if( strlen($hex_str) == 6 ) {
$color_val = hexdec($hex_str);
$rgb_array['r'] = 0xFF & ($color_val >> 0x10);
$rgb_array['g'] = 0xFF & ($color_val >> 0x8);
$rgb_array['b'] = 0xFF & $color_val;
} elseif( strlen($hex_str) == 3 ) {
$rgb_array['r'] = hexdec(str_repeat(substr($hex_str, 0, 1), 2));
$rgb_array['g'] = hexdec(str_repeat(substr($hex_str, 1, 1), 2));
$rgb_array['b'] = hexdec(str_repeat(substr($hex_str, 2, 1), 2));
} else {
return false;
}
return $return_string ? implode($separator, $rgb_array) : $rgb_array;
}
}

// Draw the image
if( isset($_GET['_CAPTCHA']) ) {

session_start();

$captcha_config = unserialize($_SESSION['_CAPTCHA']['config']);
if( !$captcha_config ) exit();

unset($_SESSION['_CAPTCHA']);

// Use milliseconds instead of seconds
srand(microtime() * 100);

// Pick random background, get info, and start captcha
$background = $captcha_config['backgrounds'][rand(0, count($captcha_config['backgrounds']) -
1)];
list($bg_width, $bg_height, $bg_type, $bg_attr) = getimagesize($background);

$captcha = imagecreatefrompng($background);

$color = hex2rgb($captcha_config['color']);
```

```
$color = imagecolorallocate($captcha, $color['r'], $color['g'], $color['b']);

// Determine text angle
$angle = rand( $captcha_config['angle_min'], $captcha_config['angle_max'] ) * (rand(0, 1) == 1 ? -
1 : 1);

// Select font randomly
$font = $captcha_config['fonts'][rand(0, count($captcha_config['fonts']) - 1)];

// Verify font file exists
if( !file_exists($font) ) throw new Exception('Font file not found: ' . $font);

//Set the font size.
$font_size = rand($captcha_config['min_font_size'], $captcha_config['max_font_size']);
$text_box_size = imagettfbbox($font_size, $angle, $font, $captcha_config['code']);

// Determine text position
$box_width = abs($text_box_size[6] - $text_box_size[2]);
$box_height = abs($text_box_size[5] - $text_box_size[1]);
$text_pos_x_min = 0;
$text_pos_x_max = ($bg_width) - ($box_width);
$text_pos_x = rand($text_pos_x_min, $text_pos_x_max);
$text_pos_y_min = $box_height;
$text_pos_y_max = ($bg_height) - ($box_height / 2);
$text_pos_y = rand($text_pos_y_min, $text_pos_y_max);

// Draw shadow
if( $captcha_config['shadow'] ){
$shadow_color = hex2rgb($captcha_config['shadow_color']);
 $shadow_color = imagecolorallocate($captcha, $shadow_color['r'], $shadow_color['g'],
$shadow_color['b']);
imagettftext($captcha, $font_size, $angle, $text_pos_x + $captcha_config['shadow_offset_x'],
$text_pos_y + $captcha_config['shadow_offset_y'], $shadow_color, $font,
$captcha_config['code']);
}

// Draw text
imagettftext($captcha, $font_size, $angle, $text_pos_x, $text_pos_y, $color, $font,
$captcha_config['code']);

// Output image
header("Content-type: image/png");
```

```
imagepng($captcha);

}

63:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\fire\fb.php
<?php

/* ***** BEGIN LICENSE BLOCK *****
 *
 * This file is part of FirePHP (http://www.firephp.org/).
 *
 * Software License Agreement (New BSD License)
 *
 * Copyright (c) 2006-2010, Christoph Dorn
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without modification,
 * are permitted provided that the following conditions are met:
 *
 *     * Redistributions of source code must retain the above copyright notice,
 *       this list of conditions and the following disclaimer.
 *
 *     * Redistributions in binary form must reproduce the above copyright notice,
 *       this list of conditions and the following disclaimer in the documentation
 *       and/or other materials provided with the distribution.
 *
 *     * Neither the name of Christoph Dorn nor the names of its
 *       contributors may be used to endorse or promote products derived from this
 *       software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
LIABLE FOR
 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
```

```
AND ON
 * ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 * ***** END LICENSE BLOCK *****
 *
 * @copyright   Copyright (C) 2007-2009 Christoph Dorn
 * @author      Christoph Dorn <christoph@christophdorn.com>
 * @license     http://www.opensource.org/licenses/bsd-license.php
 * @package     FirePHPCore
 */

if(!class_exists('FirePHP')) {
    require_once dirname(__FILE__) . DIRECTORY_SEPARATOR . 'FirePHP.class.php';
}

/**
 * Sends the given data to the FirePHP Firefox Extension.
 * The data can be displayed in the Firebug Console or in the
 * "Server" request tab.
 *
 * @see http://www.firephp.org/Wiki/Reference/Fb
 * @param mixed $Object
 * @return true
 * @throws Exception
 */
function fb()
{
    $instance = FirePHP::getInstance(true);

    $args = func_get_args();
    return call_user_func_array(array($instance,'fb'),$args);
}


class FB
{
    /**
     * Enable and disable logging to Firebug
     *
     * @see FirePHP->setEnabled()
     * @param boolean $Enabled TRUE to enable, FALSE to disable
     * @return void
     */
    public static function setEnabled($Enabled)
    {
        $instance = FirePHP::getInstance(true);
        $instance->setEnabled($Enabled);
    }

    /**
     * Check if logging is enabled
     *
     * @see FirePHP->getEnabled()
     * @return boolean TRUE if enabled
     */
    public static function getEnabled()
    {
        $instance = FirePHP::getInstance(true);
        return $instance->getEnabled();
    }

    /**
     * Specify a filter to be used when encoding an object
     *
     * Filters are used to exclude object members.
     *
     * @see FirePHP->setObjectFilter()
     * @param string $Class The class name of the object
     * @param array $Filter An array or members to exclude
     * @return void
     */
    public static function setObjectFilter($Class, $Filter)
    {
      $instance = FirePHP::getInstance(true);
      $instance->setObjectFilter($Class, $Filter);
    }

    /**
     * Set some options for the library
     *
     * @see FirePHP->setOptions()
     * @param array $Options The options to be set
     * @return void
     */
    public static function setOptions($Options)
    {
        $instance = FirePHP::getInstance(true);
        $instance->setOptions($Options);
    }

    /**
     * Get options for the library
     *
     * @see FirePHP->getOptions()
     * @return array The options
     */
    public static function getOptions()
    {
        $instance = FirePHP::getInstance(true);
        return $instance->getOptions();
    }

    /**
     * Log object to firebug
     *
     * @see http://www.firephp.org/Wiki/Reference/Fb
     * @param mixed $Object
     * @return true
     * @throws Exception
     */
    public static function send()
    {
        $instance = FirePHP::getInstance(true);
        $args = func_get_args();
        return call_user_func_array(array($instance,'fb'),$args);
    }

    /**
     * Start a group for following messages
     *
     * Options:
     *   Collapsed: [true|false]
     *   Color:     [#RRGGBB|ColorName]
     *
     * @param string $Name
     * @param array $Options OPTIONAL Instructions on how to log the group
     * @return true
     */
    public static function group($Name, $Options=null)
    {
        $instance = FirePHP::getInstance(true);
        return $instance->group($Name, $Options);
    }

    /**
     * Ends a group you have started before
     *
     * @return true
     * @throws Exception
     */
    public static function groupEnd()
    {
        return self::send(null, null, FirePHP::GROUP_END);
    }

    /**
     * Log object with label to firebug console
     *
     * @see FirePHP::LOG
     * @param mixes $Object
     * @param string $Label
     * @return true
     * @throws Exception
     */
    public static function log($Object, $Label=null)
    {
        return self::send($Object, $Label, FirePHP::LOG);
    }

    /**
     * Log object with label to firebug console
     *
     * @see FirePHP::INFO
     * @param mixes $Object
     * @param string $Label
     * @return true
     * @throws Exception
     */
    public static function info($Object, $Label=null)
    {
        return self::send($Object, $Label, FirePHP::INFO);
    }

    /**
     * Log object with label to firebug console
     *
     * @see FirePHP::WARN
     * @param mixes $Object
     * @param string $Label
     * @return true
     * @throws Exception
     */
    public static function warn($Object, $Label=null)
    {
        return self::send($Object, $Label, FirePHP::WARN);
    }

    /**
     * Log object with label to firebug console
     *
     * @see FirePHP::ERROR
     * @param mixes $Object
     * @param string $Label
     * @return true
     * @throws Exception
     */
    public static function error($Object, $Label=null)
    {
        return self::send($Object, $Label, FirePHP::ERROR);
    }

    /**
     * Dumps key and variable to firebug server panel
     *
     * @see FirePHP::DUMP
     * @param string $Key
     * @param mixed $Variable
```

```
     * @return true
     * @throws Exception
     */
    public static function dump($Key, $Variable)
    {
        return self::send($Variable, $Key, FirePHP::DUMP);
    }

    /**
     * Log a trace in the firebug console
     *
     * @see FirePHP::TRACE
     * @param string $Label
     * @return true
     * @throws Exception
     */
    public static function trace($Label)
    {
        return self::send($Label, FirePHP::TRACE);
    }

    /**
     * Log a table in the firebug console
     *
     * @see FirePHP::TABLE
     * @param string $Label
     * @param string $Table
     * @return true
     * @throws Exception
     */
    public static function table($Label, $Table)
    {
        return self::send($Table, $Label, FirePHP::TABLE);
    }

}

64:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\fire\FirePHP.class.php
<?php
/**
 * *** BEGIN LICENSE BLOCK *****
 *
```

```
 * @author         Christoph Dorn <christoph@christophdorn.com>
 * @license        http://www.opensource.org/licenses/bsd-license.php
 * @package        FirePHPCore
 */

/**
 * @see http://code.google.com/p/firephp/issues/detail?id=112
 */
if (!defined('E_STRICT')) {
    define('E_STRICT', 2048);
}
if (!defined('E_RECOVERABLE_ERROR')) {
    define('E_RECOVERABLE_ERROR', 4096);
}
if (!defined('E_DEPRECATED')) {
    define('E_DEPRECATED', 8192);
}
if (!defined('E_USER_DEPRECATED')) {
    define('E_USER_DEPRECATED', 16384);
}

/**
 * Sends the given data to the FirePHP Firefox Extension.
 * The data can be displayed in the Firebug Console or in the
 * "Server" request tab.
 *
 * For more information see: http://www.firephp.org/
 *
 * @copyright      Copyright (C) 2007-2009 Christoph Dorn
 * @author         Christoph Dorn <christoph@christophdorn.com>
 * @license        http://www.opensource.org/licenses/bsd-license.php
 * @package        FirePHPCore
 */
class FirePHP {

    /**
     * FirePHP version
     *
     * @var string
     */
    const VERSION = '0.3';    // @pinf replace '0.3' with '%%package.version%%'
    /**
     * Firebug LOG level
     *
     * Logs a message to firebug console.
     *
     * @var string
     */
    const LOG = 'LOG';

    /**
     * Firebug INFO level
     *
     * Logs a message to firebug console and displays an info icon before the message.
     *
     * @var string
     */
    const INFO = 'INFO';

    /**
     * Firebug WARN level
     *
     * Logs a message to firebug console, displays an warning icon before the message and colors
the line turquoise.
     *
     * @var string
     */
    const WARN = 'WARN';

    /**
     * Firebug ERROR level
     *
     * Logs a message to firebug console, displays an error icon before the message and colors the
line yellow. Also increments the firebug error count.
     *
     * @var string
     */
    const ERROR = 'ERROR';

    /**
     * Dumps a variable to firebug's server panel
     *
     * @var string
```

```
 */
const DUMP = 'DUMP';

/**
 * Displays a stack trace in firebug console
 *
 * @var string
 */
const TRACE = 'TRACE';

/**
 * Displays an exception in firebug console
 *
 * Increments the firebug error count.
 *
 * @var string
 */
const EXCEPTION = 'EXCEPTION';

/**
 * Displays an table in firebug console
 *
 * @var string
 */
const TABLE = 'TABLE';

/**
 * Starts a group in firebug console
 *
 * @var string
 */
const GROUP_START = 'GROUP_START';

/**
 * Ends a group in firebug console
 *
 * @var string
 */
const GROUP_END = 'GROUP_END';

/**
 * Singleton instance of FirePHP
 *
 * @var FirePHP
 */
protected static $instance = null;

/**
 * Flag whether we are logging from within the exception handler
 *
 * @var boolean
 */
protected $inExceptionHandler = false;

/**
 * Flag whether to throw PHP errors that have been converted to ErrorExceptions
 *
 * @var boolean
 */
protected $throwErrorExceptions = true;

/**
 * Flag whether to convert PHP assertion errors to Exceptions
 *
 * @var boolean
 */
protected $convertAssertionErrorsToExceptions = true;

/**
 * Flag whether to throw PHP assertion errors that have been converted to Exceptions
 *
 * @var boolean
 */
protected $throwAssertionExceptions = false;

/**
 * Wildfire protocol message index
 *
 * @var int
 */
protected $messageIndex = 1;

/**
 * Options for the library
 *
 * @var array
 */
protected $options = array('maxDepth' => 10,
                           'maxObjectDepth' => 5,
                           'maxArrayDepth' => 5,
                           'useNativeJsonEncode' => true,
                           'includeLineNumbers' => true);

/**
 * Filters used to exclude object members when encoding
 *
 * @var array
 */
protected $objectFilters = array(
    'firephp' => array('objectStack', 'instance', 'json_objectStack'),
    'firephp_test_class' => array('objectStack', 'instance', 'json_objectStack')
);

/**
 * A stack of objects used to detect recursion during object encoding
 *
 * @var object
 */
protected $objectStack = array();

/**
 * Flag to enable/disable logging
 *
 * @var boolean
 */
protected $enabled = true;

/**
 * The insight console to log to if applicable
 *
 * @var object
 */
protected $logToInsightConsole = null;

/**
 * When the object gets serialized only include specific object members.
 *
 * @return array
 */
public function __sleep()
{
    return array('options','objectFilters','enabled');
}

/**
 * Gets singleton instance of FirePHP
 *
 * @param boolean $AutoCreate
 * @return FirePHP
 */
public static function getInstance($AutoCreate = false)
{
    if ($AutoCreate===true && !self::$instance) {
        self::init();
    }
    return self::$instance;
}

/**
 * Creates FirePHP object and stores it for singleton access
 *
 * @return FirePHP
 */
public static function init()
{
    return self::setInstance(new self());
}

/**
 * Set the instance of the FirePHP singleton
 *
 * @param FirePHP $instance The FirePHP object instance
 * @return FirePHP
 */
public static function setInstance($instance)
{
    return self::$instance = $instance;
}
```

```
    /**
     * Set an Insight console to direct all logging calls to
     *
     * @param object $console The console object to log to
     * @return void
     */
    public function setLogToInsightConsole($console)
    {
        if(is_string($console)) {
            if(get_class($this)!='FirePHP_Insight' && !is_subclass_of($this, 'FirePHP_Insight')) {
                throw new Exception('FirePHP instance not an instance or subclass of
FirePHP_Insight!');
            }
            $this->logToInsightConsole = $this->to('request')->console($console);
        } else {
            $this->logToInsightConsole = $console;
        }
    }

    /**
     * Enable and disable logging to Firebug
     *
     * @param boolean $Enabled TRUE to enable, FALSE to disable
     * @return void
     */
    public function setEnabled($Enabled)
    {
       $this->enabled = $Enabled;
    }

    /**
     * Check if logging is enabled
     *
     * @return boolean TRUE if enabled
     */
    public function getEnabled()
    {
        return $this->enabled;
    }

    /**
```

```
 * Specify a filter to be used when encoding an object
 *
 * Filters are used to exclude object members.
 *
 * @param string $Class The class name of the object
 * @param array $Filter An array of members to exclude
 * @return void
 */
public function setObjectFilter($Class, $Filter)
{
    $this->objectFilters[strtolower($Class)] = $Filter;
}

/**
 * Set some options for the library
 *
 * Options:
 *  - maxDepth: The maximum depth to traverse (default: 10)
 *  - maxObjectDepth: The maximum depth to traverse objects (default: 5)
 *  - maxArrayDepth: The maximum depth to traverse arrays (default: 5)
 *  - useNativeJsonEncode: If true will use json_encode() (default: true)
 *  - includeLineNumbers: If true will include line numbers and filenames (default: true)
 *
 * @param array $Options The options to be set
 * @return void
 */
public function setOptions($Options)
{
    $this->options = array_merge($this->options,$Options);
}

/**
 * Get options from the library
 *
 * @return array The currently set options
 */
public function getOptions()
{
    return $this->options;
}

/**
```

```
 * Set an option for the library
 *
 * @param string $Name
 * @param mixed $Value
 * @throws Exception
 * @return void
 */
public function setOption($Name, $Value)
{
    if (!isset($this->options[$Name])) {
        throw $this->newException('Unknown option: ' . $Name);
    }
    $this->options[$Name] = $Value;
}

/**
 * Get an option from the library
 *
 * @param string $Name
 * @throws Exception
 * @return mixed
 */
public function getOption($Name)
{
    if (!isset($this->options[$Name])) {
        throw $this->newException('Unknown option: ' . $Name);
    }
    return $this->options[$Name];
}

/**
 * Register FirePHP as your error handler
 *
 * Will throw exceptions for each php error.
 *
 * @return mixed Returns a string containing the previously defined error handler (if any)
 */
public function registerErrorHandler($throwErrorExceptions = false)
{
    //NOTE: The following errors will not be caught by this error handler:
    //      E_ERROR, E_PARSE, E_CORE_ERROR,
    //      E_CORE_WARNING, E_COMPILE_ERROR,
    //      E_COMPILE_WARNING, E_STRICT

    $this->throwErrorExceptions = $throwErrorExceptions;

    return set_error_handler(array($this,'errorHandler'));
}

/**
 * FirePHP's error handler
 *
 * Throws exception for each php error that will occur.
 *
 * @param int $errno
 * @param string $errstr
 * @param string $errfile
 * @param int $errline
 * @param array $errcontext
 */
public function errorHandler($errno, $errstr, $errfile, $errline, $errcontext)
{
    // Don't throw exception if error reporting is switched off
    if (error_reporting() == 0) {
        return;
    }
    // Only throw exceptions for errors we are asking for
    if (error_reporting() & $errno) {

        $exception = new ErrorException($errstr, 0, $errno, $errfile, $errline);
        if ($this->throwErrorExceptions) {
            throw $exception;
        } else {
            $this->fb($exception);
        }
    }
}

/**
 * Register FirePHP as your exception handler
 *
 * @return mixed Returns the name of the previously defined exception handler,
 *              or NULL on error.
 *              If no previous handler was defined, NULL is also returned.
```

```
     */
    public function registerExceptionHandler()
    {
        return set_exception_handler(array($this,'exceptionHandler'));
    }

    /**
     * FirePHP's exception handler
     *
     * Logs all exceptions to your firebug console and then stops the script.
     *
     * @param Exception $Exception
     * @throws Exception
     */
    function exceptionHandler($Exception)
    {

        $this->inExceptionHandler = true;

        header('HTTP/1.1 500 Internal Server Error');

        try {
            $this->fb($Exception);
        } catch (Exception $e) {
            echo 'We had an exception: ' . $e;
        }
        $this->inExceptionHandler = false;
    }

    /**
     * Register FirePHP driver as your assert callback
     *
     * @param boolean $convertAssertionErrorsToExceptions
     * @param boolean $throwAssertionExceptions
     * @return mixed Returns the original setting or FALSE on errors
     */
    public function registerAssertionHandler($convertAssertionErrorsToExceptions = true,
$throwAssertionExceptions = false)
    {
        $this->convertAssertionErrorsToExceptions = $convertAssertionErrorsToExceptions;
        $this->throwAssertionExceptions = $throwAssertionExceptions;
        if ($throwAssertionExceptions && !$convertAssertionErrorsToExceptions) {
            throw $this->newException('Cannot throw assertion exceptions as assertion errors are not
being converted to exceptions!');
        }

        return assert_options(ASSERT_CALLBACK, array($this, 'assertionHandler'));
    }

    /**
     * FirePHP's assertion handler
     *
     * Logs all assertions to your firebug console and then stops the script.
     *
     * @param string $file File source of assertion
     * @param int    $line Line source of assertion
     * @param mixed  $code Assertion code
     */
    public function assertionHandler($file, $line, $code)
    {
        if ($this->convertAssertionErrorsToExceptions) {

          $exception = new ErrorException('Assertion Failed - Code[ '.$code.' ]', 0, null, $file, $line);

          if ($this->throwAssertionExceptions) {
              throw $exception;
          } else {
              $this->fb($exception);
          }

        } else {
            $this->fb($code, 'Assertion Failed', FirePHP::ERROR, array('File'=>$file,'Line'=>$line));
        }
    }

    /**
     * Start a group for following messages.
     *
     * Options:
     *   Collapsed: [true|false]
     *   Color:     [#RRGGBB|ColorName]
     *
     * @param string $Name
```

```
 * @param array $Options OPTIONAL Instructions on how to log the group
 * @return true
 * @throws Exception
 */
public function group($Name, $Options = null)
{

    if (!$Name) {
        throw $this->newException('You must specify a label for the group!');
    }

    if ($Options) {
        if (!is_array($Options)) {
            throw $this->newException('Options must be defined as an array!');
        }
        if (array_key_exists('Collapsed', $Options)) {
            $Options['Collapsed'] = ($Options['Collapsed'])?'true':'false';
        }
    }

    return $this->fb(null, $Name, FirePHP::GROUP_START, $Options);
}

/**
 * Ends a group you have started before
 *
 * @return true
 * @throws Exception
 */
public function groupEnd()
{
    return $this->fb(null, null, FirePHP::GROUP_END);
}

/**
 * Log object with label to firebug console
 *
 * @see FirePHP::LOG
 * @param mixes $Object
 * @param string $Label
 * @return true
 * @throws Exception
 */
public function log($Object, $Label = null, $Options = array())
{
    return $this->fb($Object, $Label, FirePHP::LOG, $Options);
}

/**
 * Log object with label to firebug console
 *
 * @see FirePHP::INFO
 * @param mixes $Object
 * @param string $Label
 * @return true
 * @throws Exception
 */
public function info($Object, $Label = null, $Options = array())
{
    return $this->fb($Object, $Label, FirePHP::INFO, $Options);
}

/**
 * Log object with label to firebug console
 *
 * @see FirePHP::WARN
 * @param mixes $Object
 * @param string $Label
 * @return true
 * @throws Exception
 */
public function warn($Object, $Label = null, $Options = array())
{
    return $this->fb($Object, $Label, FirePHP::WARN, $Options);
}

/**
 * Log object with label to firebug console
 *
 * @see FirePHP::ERROR
 * @param mixes $Object
 * @param string $Label
 * @return true
 * @throws Exception
```

```
     */
    public function error($Object, $Label = null, $Options = array())
    {
        return $this->fb($Object, $Label, FirePHP::ERROR, $Options);
    }

    /**
     * Dumps key and variable to firebug server panel
     *
     * @see FirePHP::DUMP
     * @param string $Key
     * @param mixed $Variable
     * @return true
     * @throws Exception
     */
    public function dump($Key, $Variable, $Options = array())
    {
        if (!is_string($Key)) {
            throw $this->newException('Key passed to dump() is not a string');
        }
        if (strlen($Key)>100) {
            throw $this->newException('Key passed to dump() is longer than 100 characters');
        }
        if (!preg_match_all('/^[a-zA-Z0-9-_\.:]*$/', $Key, $m)) {
            throw $this->newException('Key passed to dump() contains invalid characters [a-zA-Z0-9-
_\.:]');
        }
        return $this->fb($Variable, $Key, FirePHP::DUMP, $Options);
    }

    /**
     * Log a trace in the firebug console
     *
     * @see FirePHP::TRACE
     * @param string $Label
     * @return true
     * @throws Exception
     */
    public function trace($Label)
    {
        return $this->fb($Label, FirePHP::TRACE);
    }
```

```
/**
 * Log a table in the firebug console
 *
 * @see FirePHP::TABLE
 * @param string $Label
 * @param string $Table
 * @return true
 * @throws Exception
 */
public function table($Label, $Table, $Options = array())
{
    return $this->fb($Table, $Label, FirePHP::TABLE, $Options);
}

/**
 * Insight API wrapper
 *
 * @see Insight_Helper::to()
 */
public static function to()
{
    $instance = self::getInstance();
    if (!method_exists($instance, "_to")) {
        throw new Exception("FirePHP::to() implementation not loaded");
    }
    $args = func_get_args();
    return call_user_func_array(array($instance, '_to'), $args);
}

/**
 * Insight API wrapper
 *
 * @see Insight_Helper::plugin()
 */
public static function plugin()
{
    $instance = self::getInstance();
    if (!method_exists($instance, "_plugin")) {
        throw new Exception("FirePHP::plugin() implementation not loaded");
    }
    $args = func_get_args();
```

```
        return call_user_func_array(array($instance, '_plugin'), $args);
    }

    /**
     * Check if FirePHP is installed on client
     *
     * @return boolean
     */
    public function detectClientExtension()
    {
        // Check if FirePHP is installed on client via User-Agent header
        if (@preg_match_all('/\sFirePHP\/([\.\d]*)\s?/si',$this->getUserAgent(),$m) &&
           version_compare($m[1][0],'0.0.6','>=')) {
            return true;
        } else
        // Check if FirePHP is installed on client via X-FirePHP-Version header
        if (@preg_match_all('/^([\.\d]*)$/si',$this->getRequestHeader("X-FirePHP-Version"),$m) &&
           version_compare($m[1][0],'0.0.6','>=')) {
            return true;
        }
        return false;
    }

    /**
     * Log varible to Firebug
     *
     * @see http://www.firephp.org/Wiki/Reference/Fb
     * @param mixed $Object The variable to be logged
     * @return true Return TRUE if message was added to headers, FALSE otherwise
     * @throws Exception
     */
    public function fb($Object)
    {
        if($this instanceof FirePHP_Insight && method_exists($this, '_logUpgradeClientMessage')) {
            if(!FirePHP_Insight::$upgradeClientMessageLogged) {    // avoid infinite recursion as
_logUpgradeClientMessage() logs a message
                $this->_logUpgradeClientMessage();
            }
        }

        static $insightGroupStack = array();
        if (!$this->getEnabled()) {
            return false;
        }

        if ($this->headersSent($filename, $linenum)) {
            // If we are logging from within the exception handler we cannot throw another exception
            if ($this->inExceptionHandler) {
                // Simply echo the error out to the page
                echo '<div style="border: 2px solid red; font-family: Arial; font-size: 12px; background-
color: lightgray; padding: 5px;"><span style="color: red; font-weight: bold;">FirePHP
ERROR:</span> Headers already sent in <b>'.$filename.'</b> on line <b>'.$linenum.'</b>.
Cannot send log data to FirePHP. You must have Output Buffering enabled via ob_start() or
output_buffering ini directive.</div>';
            } else {
                throw $this->newException('Headers already sent in '.$filename.' on line '.$linenum.'.
Cannot send log data to FirePHP. You must have Output Buffering enabled via ob_start() or
output_buffering ini directive.');
            }
        }

        $Type = null;
        $Label = null;
        $Options = array();

        if (func_num_args()==1) {
        } else
        if (func_num_args()==2) {
            switch(func_get_arg(1)) {
                case self::LOG:
                case self::INFO:
                case self::WARN:
                case self::ERROR:
                case self::DUMP:
                case self::TRACE:
                case self::EXCEPTION:
                case self::TABLE:
                case self::GROUP_START:
                case self::GROUP_END:
                    $Type = func_get_arg(1);
                    break;
                default:
                    $Label = func_get_arg(1);
                break;
            }
        } else
        if (func_num_args()==3) {
            $Type = func_get_arg(2);
            $Label = func_get_arg(1);
        } else
        if (func_num_args()==4) {
            $Type = func_get_arg(2);
            $Label = func_get_arg(1);
            $Options = func_get_arg(3);
        } else {
            throw $this->newException('Wrong number of arguments to fb() function!');
        }

        if($this->logToInsightConsole!==null && (get_class($this)=='FirePHP_Insight' ||
is_subclass_of($this, 'FirePHP_Insight'))) {
            $msg = $this->logToInsightConsole;
            if ($Object instanceof Exception) {
                $Type = self::EXCEPTION;
            }
            if($Label && $Type!=self::TABLE && $Type!=self::GROUP_START) {
                $msg = $msg->label($Label);
            }
            switch($Type) {
                case self::DUMP:
                case self::LOG:
                    return $msg->log($Object);
                case self::INFO:
                    return $msg->info($Object);
                case self::WARN:
                    return $msg->warn($Object);
                case self::ERROR:
                    return $msg->error($Object);
                case self::TRACE:
                    return $msg->trace($Object);
                case self::EXCEPTION:
                return $this->plugin('engine')->handleException($Object, $msg);
                case self::TABLE:
                    if (isset($Object[0]) && !is_string($Object[0]) && $Label) {
                        $Object = array($Label, $Object);
                    }
            return $msg->table($Object[0], array_slice($Object[1],1), $Object[1][0]);
          case self::GROUP_START:
          $insightGroupStack[] = $msg->group(md5($Label))->open();
            return $msg->log($Label);
          case self::GROUP_END:
          if(count($insightGroupStack)==0) {
            throw new Error('Too many groupEnd() as opposed to group() calls!');
          }
          $group = array_pop($insightGroupStack);
            return $group->close();
        default:
            return $msg->log($Object);
        }
      }

      if (!$this->detectClientExtension()) {
        return false;
      }

      $meta = array();
      $skipFinalObjectEncode = false;

      if ($Object instanceof Exception) {

        $meta['file'] = $this->_escapeTraceFile($Object->getFile());
        $meta['line'] = $Object->getLine();

        $trace = $Object->getTrace();
        if ($Object instanceof ErrorException
          && isset($trace[0]['function'])
          && $trace[0]['function']=='errorHandler'
          && isset($trace[0]['class'])
          && $trace[0]['class']=='FirePHP') {

          $severity = false;
          switch($Object->getSeverity()) {
            case E_WARNING: $severity = 'E_WARNING'; break;
            case E_NOTICE: $severity = 'E_NOTICE'; break;
            case E_USER_ERROR: $severity = 'E_USER_ERROR'; break;
            case E_USER_WARNING: $severity = 'E_USER_WARNING'; break;
            case E_USER_NOTICE: $severity = 'E_USER_NOTICE'; break;
            case E_STRICT: $severity = 'E_STRICT'; break;
```

```
                case E_RECOVERABLE_ERROR: $severity = 'E_RECOVERABLE_ERROR'; break;
                case E_DEPRECATED: $severity = 'E_DEPRECATED'; break;
                case E_USER_DEPRECATED: $severity = 'E_USER_DEPRECATED'; break;
            }

            $Object = array('Class'=>get_class($Object),
                            'Message'=>$severity.': '.$Object->getMessage(),
                            'File'=>$this->_escapeTraceFile($Object->getFile()),
                            'Line'=>$Object->getLine(),
                            'Type'=>'trigger',
                            'Trace'=>$this->_escapeTrace(array_splice($trace,2)));
            $skipFinalObjectEncode = true;
        } else {
            $Object = array('Class'=>get_class($Object),
                            'Message'=>$Object->getMessage(),
                            'File'=>$this->_escapeTraceFile($Object->getFile()),
                            'Line'=>$Object->getLine(),
                            'Type'=>'throw',
                            'Trace'=>$this->_escapeTrace($trace));
            $skipFinalObjectEncode = true;
        }
        $Type = self::EXCEPTION;

    } else
    if ($Type==self::TRACE) {

        $trace = debug_backtrace();
        if (!$trace) return false;
        for( $i=0 ; $i<sizeof($trace) ; $i++ ) {

            if (isset($trace[$i]['class'])
               && isset($trace[$i]['file'])
               && ($trace[$i]['class']=='FirePHP'
                   || $trace[$i]['class']=='FB')
               && (substr($this->_standardizePath($trace[$i]['file']),-18,18)=='FirePHPCore/fb.php'
                   || substr($this->_standardizePath($trace[$i]['file']),-
29,29)=='FirePHPCore/FirePHP.class.php')) {
                /* Skip - FB::trace(), FB::send(), $firephp->trace(), $firephp->fb() */
            } else
            if (isset($trace[$i]['class'])
               && isset($trace[$i+1]['file'])
               && $trace[$i]['class']=='FirePHP'
```

```
              && substr($this->_standardizePath($trace[$i+1]['file']),-18,18)=='FirePHPCore/fb.php')
{
               /* Skip fb() */
             } else
             if ($trace[$i]['function']=='fb'
               || $trace[$i]['function']=='trace'
               || $trace[$i]['function']=='send') {

               $Object = array('Class'=>isset($trace[$i]['class'])?$trace[$i]['class']:'',
                       'Type'=>isset($trace[$i]['type'])?$trace[$i]['type']:'',
                       'Function'=>isset($trace[$i]['function'])?$trace[$i]['function']:'',
                       'Message'=>$trace[$i]['args'][0],
                       'File'=>isset($trace[$i]['file'])?$this->_escapeTraceFile($trace[$i]['file']):'',
                       'Line'=>isset($trace[$i]['line'])?$trace[$i]['line']:'',
                       'Args'=>isset($trace[$i]['args'])?$this->encodeObject($trace[$i]['args']):'',
                       'Trace'=>$this->_escapeTrace(array_splice($trace,$i+1)));

               $skipFinalObjectEncode = true;
               $meta['file'] = isset($trace[$i]['file'])?$this->_escapeTraceFile($trace[$i]['file']):'';
               $meta['line'] = isset($trace[$i]['line'])?$trace[$i]['line']:'';
               break;
             }
           }

         } else
         if ($Type==self::TABLE) {

           if (isset($Object[0]) && is_string($Object[0])) {
             $Object[1] = $this->encodeTable($Object[1]);
           } else {
             $Object = $this->encodeTable($Object);
           }

           $skipFinalObjectEncode = true;

         } else
         if ($Type==self::GROUP_START) {

           if (!$Label) {
             throw $this->newException('You must specify a label for the group!');
           }
```

```
        } else {
            if ($Type===null) {
                $Type = self::LOG;
            }
        }

        if ($this->options['includeLineNumbers']) {
            if (!isset($meta['file']) || !isset($meta['line'])) {

                $trace = debug_backtrace();
                for( $i=0 ; $trace && $i<sizeof($trace) ; $i++ ) {

                    if (isset($trace[$i]['class'])
                        && isset($trace[$i]['file'])
                        && ($trace[$i]['class']=='FirePHP'
                            || $trace[$i]['class']=='FB')
                        && (substr($this->_standardizePath($trace[$i]['file']),-18,18)=='FirePHPCore/fb.php'
                            || substr($this->_standardizePath($trace[$i]['file']),-
29,29)=='FirePHPCore/FirePHP.class.php')) {
                        /* Skip - FB::trace(), FB::send(), $firephp->trace(), $firephp->fb() */
                    } else
                    if (isset($trace[$i]['class'])
                        && isset($trace[$i+1]['file'])
                        && $trace[$i]['class']=='FirePHP'
                        && substr($this->_standardizePath($trace[$i+1]['file']),-
18,18)=='FirePHPCore/fb.php') {
                        /* Skip fb() */
                    } else
                    if (isset($trace[$i]['file'])
                        && substr($this->_standardizePath($trace[$i]['file']),-18,18)=='FirePHPCore/fb.php')
{
                        /* Skip FB::fb() */
                    } else {
                        $meta['file'] = isset($trace[$i]['file'])?$this->_escapeTraceFile($trace[$i]['file']):'';
                        $meta['line'] = isset($trace[$i]['line'])?$trace[$i]['line']:'';
                        break;
                    }
                }
            }
        } else {
            unset($meta['file']);
            unset($meta['line']);
        }

        $this->setHeader('X-Wf-Protocol-1','http://meta.wildfirehq.org/Protocol/JsonStream/0.2');
        $this->setHeader('X-Wf-1-Plugin-1','http://meta.firephp.org/Wildfire/Plugin/FirePHP/Library-
FirePHPCore/'.self::VERSION);

        $structure_index = 1;
        if ($Type==self::DUMP) {
            $structure_index = 2;
            $this->setHeader('X-Wf-1-Structure-
2','http://meta.firephp.org/Wildfire/Structure/FirePHP/Dump/0.1');
        } else {
            $this->setHeader('X-Wf-1-Structure-
1','http://meta.firephp.org/Wildfire/Structure/FirePHP/FirebugConsole/0.1');
        }

        if ($Type==self::DUMP) {
            $msg = '{"'.$Label.'":'.$this->jsonEncode($Object, $skipFinalObjectEncode).'}';
        } else {
            $msg_meta = $Options;
            $msg_meta['Type'] = $Type;
            if ($Label!==null) {
                $msg_meta['Label'] = $Label;
            }
            if (isset($meta['file']) && !isset($msg_meta['File'])) {
                $msg_meta['File'] = $meta['file'];
            }
            if (isset($meta['line']) && !isset($msg_meta['Line'])) {
                $msg_meta['Line'] = $meta['line'];
            }
            $msg = '['.$this->jsonEncode($msg_meta).','.$this->jsonEncode($Object,
$skipFinalObjectEncode).']';
        }

        $parts = explode("\n",chunk_split($msg, 5000, "\n"));

        for( $i=0 ; $i<count($parts) ; $i++) {

            $part = $parts[$i];
            if ($part) {

                if (count($parts)>2) {
```

```
                // Message needs to be split into multiple parts
                $this->setHeader('X-Wf-1-'.$structure_index.'-'.'1-'.$this->messageIndex,
                                 (($i==0)?strlen($msg):'')
                                 . '|' . $part . '|'
                                 . (($i<count($parts)-2)?'\\':''));
            } else {
                $this->setHeader('X-Wf-1-'.$structure_index.'-'.'1-'.$this->messageIndex,
                                 strlen($part) . '|' . $part . '|');
            }

            $this->messageIndex++;

            if ($this->messageIndex > 99999) {
                throw $this->newException('Maximum number (99,999) of messages reached!');

            }
        }
    }

    $this->setHeader('X-Wf-1-Index',$this->messageIndex-1);

    return true;
}

/**
 * Standardizes path for windows systems.
 *
 * @param string $Path
 * @return string
 */
protected function _standardizePath($Path)
{
    return preg_replace('/\\\\+/','/',$Path);
}

/**
 * Escape trace path for windows systems
 *
 * @param array $Trace
 * @return array
 */
protected function _escapeTrace($Trace)
```

```
{
    if (!$Trace) return $Trace;
    for( $i=0 ; $i<sizeof($Trace) ; $i++ ) {
        if (isset($Trace[$i]['file'])) {
            $Trace[$i]['file'] = $this->_escapeTraceFile($Trace[$i]['file']);
        }
        if (isset($Trace[$i]['args'])) {
            $Trace[$i]['args'] = $this->encodeObject($Trace[$i]['args']);
        }
    }
    return $Trace;
}

/**
 * Escape file information of trace for windows systems
 *
 * @param string $File
 * @return string
 */
protected function _escapeTraceFile($File)
{
    /* Check if we have a windows filepath */
    if (strpos($File,'\\')) {
        /* First strip down to single \ */

        $file = preg_replace('/\\\\+/','\\',$File);

        return $file;
    }
    return $File;
}

/**
 * Check if headers have already been sent
 *
 * @param string $Filename
 * @param integer $Linenum
 */
protected function headersSent(&$Filename, &$Linenum)
{
    return headers_sent($Filename, $Linenum);
}
```

```
/**
 * Send header
 *
 * @param string $Name
 * @param string $Value
 */
protected function setHeader($Name, $Value)
{
    return header($Name.': '.$Value);
}

/**
 * Get user agent
 *
 * @return string|false
 */
protected function getUserAgent()
{
    if (!isset($_SERVER['HTTP_USER_AGENT'])) return false;
    return $_SERVER['HTTP_USER_AGENT'];
}

/**
 * Get all request headers
 *
 * @return array
 */
public static function getAllRequestHeaders() {
    static $_cached_headers = false;
    if($_cached_headers!==false) {
        return $_cached_headers;
    }
    $headers = array();
    if(function_exists('getallheaders')) {
        foreach( getallheaders() as $name => $value ) {
            $headers[strtolower($name)] = $value;
        }
    } else {
        foreach($_SERVER as $name => $value) {
            if(substr($name, 0, 5) == 'HTTP_') {
                $headers[strtolower(str_replace(' ', '-', str_replace('_', ' ', substr($name, 5))))] =
$value;
                }
            }
        }
        return $_cached_headers = $headers;
    }

    /**
     * Get a request header
     *
     * @return string|false
     */
    protected function getRequestHeader($Name)
    {
        $headers = self::getAllRequestHeaders();
        if (isset($headers[strtolower($Name)])) {
            return $headers[strtolower($Name)];
        }
        return false;
    }

    /**
     * Returns a new exception
     *
     * @param string $Message
     * @return Exception
     */
    protected function newException($Message)
    {
        return new Exception($Message);
    }

    /**
     * Encode an object into a JSON string
     *
     * Uses PHP's jeson_encode() if available
     *
     * @param object $Object The object to be encoded
     * @return string The JSON string
     */
    public function jsonEncode($Object, $skipObjectEncode = false)
    {
```

```
    if (!$skipObjectEncode) {
        $Object = $this->encodeObject($Object);
    }

    if (function_exists('json_encode')
       && $this->options['useNativeJsonEncode']!=false) {

        return json_encode($Object);
    } else {
        return $this->json_encode($Object);
    }
}

/**
 * Encodes a table by encoding each row and column with encodeObject()
 *
 * @param array $Table The table to be encoded
 * @return array
 */
protected function encodeTable($Table)
{

    if (!$Table) return $Table;

    $new_table = array();
    foreach($Table as $row) {

        if (is_array($row)) {
            $new_row = array();

            foreach($row as $item) {
                $new_row[] = $this->encodeObject($item);
            }

            $new_table[] = $new_row;
        }
    }

    return $new_table;
}

/**
```

```
 * Encodes an object including members with
 * protected and private visibility
 *
 * @param Object $Object The object to be encoded
 * @param int $Depth The current traversal depth
 * @return array All members of the object
 */
protected function encodeObject($Object, $ObjectDepth = 1, $ArrayDepth = 1, $MaxDepth = 1)
{
    if ($MaxDepth > $this->options['maxDepth']) {
        return '** Max Depth ('.$this->options['maxDepth'].') **';
    }

    $return = array();

    if (is_resource($Object)) {

        return '** '.(string)$Object.' **';

    } else
    if (is_object($Object)) {

        if ($ObjectDepth > $this->options['maxObjectDepth']) {
            return '** Max Object Depth ('.$this->options['maxObjectDepth'].') **';
        }

        foreach ($this->objectStack as $refVal) {
            if ($refVal === $Object) {
                return '** Recursion ('.get_class($Object).') **';
            }
        }
        array_push($this->objectStack, $Object);

        $return['__className'] = $class = get_class($Object);
        $class_lower = strtolower($class);

        $reflectionClass = new ReflectionClass($class);
        $properties = array();
        foreach( $reflectionClass->getProperties() as $property) {
            $properties[$property->getName()] = $property;
        }
        $members = (array)$Object;

        foreach( $properties as $plain_name => $property ) {

            $name = $raw_name = $plain_name;
            if ($property->isStatic()) {
                $name = 'static:'.$name;
            }
            if ($property->isPublic()) {
                $name = 'public:'.$name;
            } else
            if ($property->isPrivate()) {
                $name = 'private:'.$name;
                $raw_name = "\0".$class."\0".$raw_name;
            } else
            if ($property->isProtected()) {
                $name = 'protected:'.$name;
                $raw_name = "\0".'*'."\0".$raw_name;
            }

            if (!(isset($this->objectFilters[$class_lower])
                 && is_array($this->objectFilters[$class_lower])
                 && in_array($plain_name,$this->objectFilters[$class_lower]))) {

                if (array_key_exists($raw_name,$members)
                  && !$property->isStatic()) {

                    $return[$name] = $this->encodeObject($members[$raw_name], $ObjectDepth + 1,
1, $MaxDepth + 1);

                } else {
                    if (method_exists($property,'setAccessible')) {
                        $property->setAccessible(true);
                        $return[$name] = $this->encodeObject($property->getValue($Object),
$ObjectDepth + 1, 1, $MaxDepth + 1);
                    } else
                    if ($property->isPublic()) {
                        $return[$name] = $this->encodeObject($property->getValue($Object),
$ObjectDepth + 1, 1, $MaxDepth + 1);
                    } else {
                        $return[$name] = '** Need PHP 5.3 to get value **';
                    }
                }
            } else {
                $return[$name] = '** Excluded by Filter **';
            }
        }

        // Include all members that are not defined in the class
        // but exist in the object
        foreach( $members as $raw_name => $value ) {

            $name = $raw_name;

            if ($name{0} == "\0") {
                $parts = explode("\0", $name);
                $name = $parts[2];
            }

            $plain_name = $name;

            if (!isset($properties[$name])) {
                $name = 'undeclared:'.$name;

                if (!(isset($this->objectFilters[$class_lower])
                     && is_array($this->objectFilters[$class_lower])
                     && in_array($plain_name,$this->objectFilters[$class_lower]))) {

                    $return[$name] = $this->encodeObject($value, $ObjectDepth + 1, 1, $MaxDepth +
1);
                } else {
                    $return[$name] = '** Excluded by Filter **';
                }
            }
        }

        array_pop($this->objectStack);

    } elseif (is_array($Object)) {

        if ($ArrayDepth > $this->options['maxArrayDepth']) {
            return '** Max Array Depth ('.$this->options['maxArrayDepth'].') **';
        }
        foreach ($Object as $key => $val) {

            // Encoding the $GLOBALS PHP array causes an infinite loop
            // if the recursion is not reset here as it contains
            // a reference to itself. This is the only way I have come up
            // with to stop infinite recursion in this case.
            if ($key=='GLOBALS'
              && is_array($val)
              && array_key_exists('GLOBALS',$val)) {
               $val['GLOBALS'] = '** Recursion (GLOBALS) **';
            }

            $return[$key] = $this->encodeObject($val, 1, $ArrayDepth + 1, $MaxDepth + 1);
        }
    } else {
        if (self::is_utf8($Object)) {
            return $Object;
        } else {
            return utf8_encode($Object);
        }
    }
    return $return;
}

/**
 * Returns true if $string is valid UTF-8 and false otherwise.
 *
 * @param mixed $str String to be tested
 * @return boolean
 */
protected static function is_utf8($str)
{
    if(function_exists('mb_detect_encoding')) {
        return (mb_detect_encoding($str) == 'UTF-8');
    }
    $c=0; $b=0;
    $bits=0;
    $len=strlen($str);
    for($i=0; $i<$len; $i++){
        $c=ord($str[$i]);
        if ($c > 128){
            if (($c >= 254)) return false;
            elseif ($c >= 252) $bits=6;
            elseif ($c >= 248) $bits=5;
            elseif ($c >= 240) $bits=4;
            elseif ($c >= 224) $bits=3;
            elseif ($c >= 192) $bits=2;
            else return false;
            if (($i+$bits) > $len) return false;
            while($bits > 1){
                $i++;
                $b=ord($str[$i]);
                if ($b < 128 || $b > 191) return false;
                $bits--;
            }
        }
    }
    return true;
}

/**
 * Converts to and from JSON format.
 *
 * JSON (JavaScript Object Notation) is a lightweight data-interchange
 * format. It is easy for humans to read and write. It is easy for machines
 * to parse and generate. It is based on a subset of the JavaScript
 * Programming Language, Standard ECMA-262 3rd Edition - December 1999.
 * This feature can also be found in  Python. JSON is a text format that is
 * completely language independent but uses conventions that are familiar
 * to programmers of the C-family of languages, including C, C++, C#, Java,
 * JavaScript, Perl, TCL, and many others. These properties make JSON an
 * ideal data-interchange language.
 *
 * This package provides a simple encoder and decoder for JSON notation. It
 * is intended for use with client-side Javascript applications that make
 * use of HTTPRequest to perform server communication functions - data can
 * be encoded into JSON notation for use in a client-side javascript, or
 * decoded from incoming Javascript requests. JSON format is native to
 * Javascript, and can be directly eval()'ed with no further parsing
 * overhead
 *
 * All strings should be in ASCII or UTF-8 format!
 *
 * LICENSE: Redistribution and use in source and binary forms, with or
```

```
     * without modification, are permitted provided that the following
     * conditions are met: Redistributions of source code must retain the
     * above copyright notice, this list of conditions and the following
     * disclaimer. Redistributions in binary form must reproduce the above
     * copyright notice, this list of conditions and the following disclaimer
     * in the documentation and/or other materials provided with the
     * distribution.
     *
     * THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESS OR IMPLIED
     * WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
     * MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
IN
     * NO EVENT SHALL CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
     * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
     * BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS
     * OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
     * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR
     * TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE
     * USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH
     * DAMAGE.
     *
     * @category
     * @package     Services_JSON
     * @author      Michal Migurski <mike-json@teczno.com>
     * @author      Matt Knapp <mdknapp[at]gmail[dot]com>
     * @author      Brett Stimmerman <brettstimmerman[at]gmail[dot]com>
     * @author      Christoph Dorn <christoph@christophdorn.com>
     * @copyright   2005 Michal Migurski
     * @version     CVS: $Id: JSON.php,v 1.31 2006/06/28 05:54:17 migurski Exp $
     * @license     http://www.opensource.org/licenses/bsd-license.php
     * @link        http://pear.php.net/pepr/pepr-proposal-show.php?id=198
     */


    /**
     * Keep a list of objects as we descend into the array so we can detect recursion.
     */
    private $json_objectStack = array();


   /**
    * convert a string from one UTF-8 char to one UTF-16 char
```

```
*
* Normally should be handled by mb_convert_encoding, but
* provides a slower PHP-only method for installations
* that lack the multibye string extension.
*
* @param    string  $utf8   UTF-8 character
* @return   string  UTF-16 character
* @access   private
*/
private function json_utf82utf16($utf8)
{
    // oh please oh please oh please oh please oh please
    if (function_exists('mb_convert_encoding')) {
        return mb_convert_encoding($utf8, 'UTF-16', 'UTF-8');
    }

    switch(strlen($utf8)) {
        case 1:
            // this case should never be reached, because we are in ASCII range
            // see: http://www.cl.cam.ac.uk/~mgk25/unicode.html#utf-8
            return $utf8;

        case 2:
            // return a UTF-16 character from a 2-byte UTF-8 char
            // see: http://www.cl.cam.ac.uk/~mgk25/unicode.html#utf-8
            return chr(0x07 & (ord($utf8{0}) >> 2))
                 . chr((0xC0 & (ord($utf8{0}) << 6))
                 | (0x3F & ord($utf8{1})));

        case 3:
            // return a UTF-16 character from a 3-byte UTF-8 char
            // see: http://www.cl.cam.ac.uk/~mgk25/unicode.html#utf-8
            return chr((0xF0 & (ord($utf8{0}) << 4))
                 | (0x0F & (ord($utf8{1}) >> 2)))
                 . chr((0xC0 & (ord($utf8{1}) << 6))
                 | (0x7F & ord($utf8{2})));
    }

    // ignoring UTF-32 for now, sorry
    return '';
}
```

```
/**
 * encodes an arbitrary variable into JSON format
 *
 * @param    mixed   $var    any number, boolean, string, array, or object to be encoded.
 *                           see argument 1 to Services_JSON() above for array-parsing behavior.
 *                           if var is a strng, note that encode() always expects it
 *                           to be in ASCII or UTF-8 format!
 *
 * @return   mixed   JSON string representation of input var or an error if a problem occurs
 * @access   public
 */
 private function json_encode($var)
 {

     if (is_object($var)) {
         if (in_array($var,$this->json_objectStack)) {
             return '"** Recursion **"';
         }
     }

     switch (gettype($var)) {
         case 'boolean':
             return $var ? 'true' : 'false';

         case 'NULL':
             return 'null';

         case 'integer':
             return (int) $var;

         case 'double':
         case 'float':
             return (float) $var;

         case 'string':
             // STRINGS ARE EXPECTED TO BE IN ASCII OR UTF-8 FORMAT
             $ascii = '';
             $strlen_var = strlen($var);

            /*
             * Iterate over every character in the string,
             * escaping with a slash or encoding to UTF-8 where necessary
```

```
*/
for ($c = 0; $c < $strlen_var; ++$c) {

    $ord_var_c = ord($var{$c});

    switch (true) {
        case $ord_var_c == 0x08:
            $ascii .= '\b';
            break;
        case $ord_var_c == 0x09:
            $ascii .= '\t';
            break;
        case $ord_var_c == 0x0A:
            $ascii .= '\n';
            break;
        case $ord_var_c == 0x0C:
            $ascii .= '\f';
            break;
        case $ord_var_c == 0x0D:
            $ascii .= '\r';
            break;

        case $ord_var_c == 0x22:
        case $ord_var_c == 0x2F:
        case $ord_var_c == 0x5C:
            // double quote, slash, slosh
            $ascii .= '\\'.$var{$c};
            break;

        case (($ord_var_c >= 0x20) && ($ord_var_c <= 0x7F)):
            // characters U-00000000 - U-0000007F (same as ASCII)
            $ascii .= $var{$c};
            break;

        case (($ord_var_c & 0xE0) == 0xC0):
            // characters U-00000080 - U-000007FF, mask 110XXXXX
            // see http://www.cl.cam.ac.uk/~mgk25/unicode.html#utf-8
            $char = pack('C*', $ord_var_c, ord($var{$c + 1}));
            $c += 1;
            $utf16 = $this->json_utf82utf16($char);
            $ascii .= sprintf('\u%04s', bin2hex($utf16));
            break;
```

```
case (($ord_var_c & 0xF0) == 0xE0):
   // characters U-00000800 - U-0000FFFF, mask 1110XXXX
   // see http://www.cl.cam.ac.uk/~mgk25/unicode.html#utf-8
   $char = pack('C*', $ord_var_c,
             ord($var{$c + 1}),
             ord($var{$c + 2}));
   $c += 2;
   $utf16 = $this->json_utf82utf16($char);
   $ascii .= sprintf('\u%04s', bin2hex($utf16));
   break;

case (($ord_var_c & 0xF8) == 0xF0):
   // characters U-00010000 - U-001FFFFF, mask 11110XXX
   // see http://www.cl.cam.ac.uk/~mgk25/unicode.html#utf-8
   $char = pack('C*', $ord_var_c,
             ord($var{$c + 1}),
             ord($var{$c + 2}),
             ord($var{$c + 3}));
   $c += 3;
   $utf16 = $this->json_utf82utf16($char);
   $ascii .= sprintf('\u%04s', bin2hex($utf16));
   break;

case (($ord_var_c & 0xFC) == 0xF8):
   // characters U-00200000 - U-03FFFFFF, mask 111110XX
   // see http://www.cl.cam.ac.uk/~mgk25/unicode.html#utf-8
   $char = pack('C*', $ord_var_c,
             ord($var{$c + 1}),
             ord($var{$c + 2}),
             ord($var{$c + 3}),
             ord($var{$c + 4}));
   $c += 4;
   $utf16 = $this->json_utf82utf16($char);
   $ascii .= sprintf('\u%04s', bin2hex($utf16));
   break;

case (($ord_var_c & 0xFE) == 0xFC):
   // characters U-04000000 - U-7FFFFFFF, mask 1111110X
   // see http://www.cl.cam.ac.uk/~mgk25/unicode.html#utf-8
   $char = pack('C*', $ord_var_c,
             ord($var{$c + 1}),
                                ord($var{$c + 2}),
                                ord($var{$c + 3}),
                                ord($var{$c + 4}),
                                ord($var{$c + 5}));
                        $c += 5;
                        $utf16 = $this->json_utf82utf16($char);
                        $ascii .= sprintf('\u%04s', bin2hex($utf16));
                        break;
                }
            }

            return '"'.$ascii.'"';

        case 'array':
            /*
             * As per JSON spec if any array key is not an integer
             * we must treat the the whole array as an object. We
             * also try to catch a sparsely populated associative
             * array with numeric keys here because some JS engines
             * will create an array with empty indexes up to
             * max_index which can cause memory issues and because
             * the keys, which may be relevant, will be remapped
             * otherwise.
             *
             * As per the ECMA and JSON specification an object may
             * have any string as a property. Unfortunately due to
             * a hole in the ECMA specification if the key is a
             * ECMA reserved word or starts with a digit the
             * parameter is only accessible using ECMAScript's
             * bracket notation.
             */

            // treat as a JSON object
            if (is_array($var) && count($var) && (array_keys($var) !== range(0, sizeof($var) - 1))) {

                $this->json_objectStack[] = $var;

                $properties = array_map(array($this, 'json_name_value'),
                                        array_keys($var),
                                        array_values($var));

                array_pop($this->json_objectStack);
                foreach($properties as $property) {
                    if ($property instanceof Exception) {
                        return $property;
                    }
                }

                return '{' . join(',', $properties) . '}';
            }

            $this->json_objectStack[] = $var;

            // treat it like a regular array
            $elements = array_map(array($this, 'json_encode'), $var);

            array_pop($this->json_objectStack);

            foreach($elements as $element) {
                if ($element instanceof Exception) {
                    return $element;
                }
            }

            return '[' . join(',', $elements) . ']';

        case 'object':
            $vars = self::encodeObject($var);

            $this->json_objectStack[] = $var;

            $properties = array_map(array($this, 'json_name_value'),
                                    array_keys($vars),
                                    array_values($vars));

            array_pop($this->json_objectStack);

            foreach($properties as $property) {
                if ($property instanceof Exception) {
                    return $property;
                }
            }
            return '{' . join(',', $properties) . '}';

        default:
            return null;
    }
}

/**
 * array-walking function for use in generating JSON-formatted name-value pairs
 *
 * @param    string  $name   name of key to use
 * @param    mixed   $value  reference to an array element to be encoded
 *
 * @return   string  JSON-formatted name-value pair, like '"name":value'
 * @access   private
 */
private function json_name_value($name, $value)
{
    // Encoding the $GLOBALS PHP array causes an infinite loop
    // if the recursion is not reset here as it contains
    // a reference to itself. This is the only way I have come up
    // with to stop infinite recursion in this case.
    if ($name=='GLOBALS'
      && is_array($value)
      && array_key_exists('GLOBALS',$value)) {
       $value['GLOBALS'] = '** Recursion **';
    }

    $encoded_value = $this->json_encode($value);

    if ($encoded_value instanceof Exception) {
        return $encoded_value;
    }

    return $this->json_encode(strval($name)) . ':' . $encoded_value;
}

/**
 * @deprecated
 */
public function setProcessorUrl($URL)
{
        trigger_error("The FirePHP::setProcessorUrl() method is no longer supported",
E_USER_DEPRECATED);
    }

    /**
     * @deprecated
     */
    public function setRendererUrl($URL)
    {
        trigger_error("The FirePHP::setRendererUrl() method is no longer supported",
E_USER_DEPRECATED);
    }
}

65:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\class.phpmailer.php
<?php
/**
 * PHPMailer - PHP email creation and transport class.
 * PHP Version 5.0.0
 * Version 5.2.7
 * @package PHPMailer
 * @link https://github.com/PHPMailer/PHPMailer/
 * @author Marcus Bointon (coolbru) <phpmailer@synchromedia.co.uk>
 * @author Jim Jagielski (jimjag) <jimjag@gmail.com>
 * @author Andy Prevost (codeworxtech) <codeworxtech@users.sourceforge.net>
 * @author Brent R. Matzelle (original founder)
 * @copyright 2013 Marcus Bointon
 * @copyright 2010 - 2012 Jim Jagielski
 * @copyright 2004 - 2009 Andy Prevost
 * @license http://www.gnu.org/copyleft/lesser.html GNU Lesser General Public License
 * @note This program is distributed in the hope that it will be useful - WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.
 */

if (version_compare(PHP_VERSION, '5.0.0', '<')) {
    exit("Sorry, PHPMailer will only run on PHP version 5 or greater!\n");
}

/**
 * PHPMailer - PHP email creation and transport class.
 * PHP Version 5.0.0
 * @package PHPMailer
 * @author Marcus Bointon (coolbru) <phpmailer@synchromedia.co.uk>
 * @author Jim Jagielski (jimjag) <jimjag@gmail.com>
 * @author Andy Prevost (codeworxtech) <codeworxtech@users.sourceforge.net>
 * @author Brent R. Matzelle (original founder)
 * @copyright 2013 Marcus Bointon
 * @copyright 2010 - 2012 Jim Jagielski
 * @copyright 2004 - 2009 Andy Prevost
 */
class PHPMailer
{
    /**
     * The PHPMailer Version number.
     * @type string
     */
    public $Version = '5.2.7';

    /**
     * Email priority.
     * Options: 1 = High, 3 = Normal, 5 = low.
     * @type int
     */
    public $Priority = 3;

    /**
     * The character set of the message.
     * @type string
     */
    public $CharSet = 'iso-8859-1';

    /**
     * The MIME Content-type of the message.
     * @type string
     */
    public $ContentType = 'text/plain';

    /**
     * The message encoding.
     * Options: "8bit", "7bit", "binary", "base64", and "quoted-printable".
     * @type string
     */
    public $Encoding = '8bit';
```

```
/**
 * Holds the most recent mailer error message.
 * @type string
 */
public $ErrorInfo = '';

/**
 * The From email address for the message.
 * @type string
 */
public $From = 'root@localhost';

/**
 * The From name of the message.
 * @type string
 */
public $FromName = 'Root User';

/**
 * The Sender email (Return-Path) of the message.
 * If not empty, will be sent via -f to sendmail or as 'MAIL FROM' in smtp mode.
 * @type string
 */
public $Sender = '';

/**
 * The Return-Path of the message.
 * If empty, it will be set to either From or Sender.
 * @type string
 */
public $ReturnPath = '';

/**
 * The Subject of the message.
 * @type string
 */
public $Subject = '';

/**
 * An HTML or plain text message body.
 * If HTML then call isHTML(true).
 * @type string
 */
public $Body = '';

/**
 * The plain-text message body.
 * This body can be read by mail clients that do not have HTML email
 * capability such as mutt & Eudora.
 * Clients that can read HTML will view the normal Body.
 * @type string
 */
public $AltBody = '';

/**
 * An iCal message part body.
 * Only supported in simple alt or alt_inline message types
 * To generate iCal events, use the bundled extras/EasyPeasyICS.php class or iCalcreator
 * @link http://sprain.ch/blog/downloads/php-class-easypeasyics-create-ical-files-with-php/
 * @link http://kigkonsult.se/iCalcreator/
 * @type string
 */
public $Ical = '';

/**
 * The complete compiled MIME message body.
 * @access protected
 * @type string
 */
protected $MIMEBody = '';

/**
 * The complete compiled MIME message headers.
 * @type string
 * @access protected
 */
protected $MIMEHeader = '';

/**
 * Extra headers that createHeader() doesn't fold in.
 * @type string
 * @access protected
 */
```

```
protected $mailHeader = '';

/**
 * Word-wrap the message body to this number of chars.
 * @type int
 */
public $WordWrap = 0;

/**
 * Which method to use to send mail.
 * Options: "mail", "sendmail", or "smtp".
 * @type string
 */
public $Mailer = 'mail';

/**
 * The path to the sendmail program.
 * @type string
 */
public $Sendmail = '/usr/sbin/sendmail';

/**
 * Whether mail() uses a fully sendmail-compatible MTA.
 * One which supports sendmail's "-oi -f" options.
 * @type bool
 */
public $UseSendmailOptions = true;

/**
 * Path to PHPMailer plugins.
 * Useful if the SMTP class is not in the PHP include path.
 * @type string
 * @deprecated Should not be needed now there is an autoloader.
 */
public $PluginDir = '';

/**
 * The email address that a reading confirmation should be sent to.
 * @type string
 */
public $ConfirmReadingTo = '';
```

```
/**
 * The hostname to use in Message-Id and Received headers
 * and as default HELO string.
 * If empty, the value returned
 * by SERVER_NAME is used or 'localhost.localdomain'.
 * @type string
 */
public $Hostname = '';

/**
 * An ID to be used in the Message-Id header.
 * If empty, a unique id will be generated.
 * @type string
 */
public $MessageID = '';

/**
 * The message Date to be used in the Date header.
 * If empty, the current date will be added.
 * @type string
 */
public $MessageDate = '';

/**
 * SMTP hosts.
 * Either a single hostname or multiple semicolon-delimited hostnames.
 * You can also specify a different port
 * for each host by using this format: [hostname:port]
 * (e.g. "smtp1.example.com:25;smtp2.example.com").
 * Hosts will be tried in order.
 * @type string
 */
public $Host = 'localhost';

/**
 * The default SMTP server port.
 * @type int
 * @Todo Why is this needed when the SMTP class takes care of it?
 */
public $Port = 25;

/**
```

```
 * The SMTP HELO of the message.
 * Default is $Hostname.
 * @type string
 * @see PHPMailer::$Hostname
 */
public $Helo = '';

/**
 * The secure connection prefix.
 * Options: "", "ssl" or "tls"
 * @type string
 */
public $SMTPSecure = '';

/**
 * Whether to use SMTP authentication.
 * Uses the Username and Password properties.
 * @type bool
 * @see PHPMailer::$Username
 * @see PHPMailer::$Password
 */
public $SMTPAuth = false;

/**
 * SMTP username.
 * @type string
 */
public $Username = '';

/**
 * SMTP password.
 * @type string
 */
public $Password = '';

/**
 * SMTP auth type.
 * Options are LOGIN (default), PLAIN, NTLM, CRAM-MD5
 * @type string
 */
public $AuthType = '';
```

```
/**
 * SMTP realm.
 * Used for NTLM auth
 * @type string
 */
public $Realm = '';

/**
 * SMTP workstation.
 * Used for NTLM auth
 * @type string
 */
public $Workstation = '';

/**
 * The SMTP server timeout in seconds.
 * @type int
 */
public $Timeout = 10;

/**
 * SMTP class debug output mode.
 * Options: 0 = off, 1 = commands, 2 = commands and data
 * @type int
 * @see SMTP::$do_debug
 */
public $SMTPDebug = 0;

/**
 * The function/method to use for debugging output.
 * Options: "echo" or "error_log"
 * @type string
 * @see SMTP::$Debugoutput
 */
public $Debugoutput = "echo";

/**
 * Whether to keep SMTP connection open after each message.
 * If this is set to true then to close the connection
 * requires an explicit call to smtpClose().
 * @type bool
 */
```

```
public $SMTPKeepAlive = false;

/**
 * Whether to split multiple to addresses into multiple messages
 * or send them all in one message.
 * @type bool
 */
public $SingleTo = false;

/**
 * Storage for addresses when SingleTo is enabled.
 * @type array
 * @todo This should really not be public
 */
public $SingleToArray = array();

/**
 * Whether to generate VERP addresses on send.
 * Only applicable when sending via SMTP.
 * @link http://en.wikipedia.org/wiki/Variable_envelope_return_path
 * @type bool
 */
public $do_verp = false;

/**
 * Whether to allow sending messages with an empty body.
 * @type bool
 */
public $AllowEmpty = false;

/**
 * The default line ending.
 * @note The default remains "\n". We force CRLF where we know
 *       it must be used via self::CRLF.
 * @type string
 */
public $LE = "\n";

/**
 * DKIM selector.
 * @type string
 */
```

```
public $DKIM_selector = '';

/**
 * DKIM Identity.
 * Usually the email address used as the source of the email
 * @type string
 */
public $DKIM_identity = '';

/**
 * DKIM passphrase.
 * Used if your key is encrypted.
 * @type string
 */
public $DKIM_passphrase = '';

/**
 * DKIM signing domain name.
 * @example 'example.com'
 * @type string
 */
public $DKIM_domain = '';

/**
 * DKIM private key file path.
 * @type string
 */
public $DKIM_private = '';

/**
 * Callback Action function name.
 *
 * The function that handles the result of the send email action.
 * It is called out by send() for each email sent.
 *
 * Value can be:
 * - 'function_name' for function names
 * - 'Class::Method' for static method calls
 * - array($object, 'Method') for calling methods on $object
 * See http://php.net/is_callable manual page for more details.
 *
 * Parameters:
```

```
 *   bool    $result        result of the send action
 *   string  $to            email address of the recipient
 *   string  $cc            cc email addresses
 *   string  $bcc           bcc email addresses
 *   string  $subject       the subject
 *   string  $body          the email body
 *   string  $from          email address of sender
 *
 * @type string
 */
public $action_function = '';

/**
 * What to use in the X-Mailer header.
 * Options: null for default, whitespace for none, or a string to use
 * @type string
 */
public $XMailer = '';

/**
 * An instance of the SMTP sender class.
 * @type SMTP
 * @access protected
 */
protected $smtp = null;

/**
 * The array of 'to' addresses.
 * @type array
 * @access protected
 */
protected $to = array();

/**
 * The array of 'cc' addresses.
 * @type array
 * @access protected
 */
protected $cc = array();

/**
 * The array of 'bcc' addresses.
 * @type array
 * @access protected
 */
protected $bcc = array();

/**
 * The array of reply-to names and addresses.
 * @type array
 * @access protected
 */
protected $ReplyTo = array();

/**
 * An array of all kinds of addresses.
 * Includes all of $to, $cc, $bcc, $replyto
 * @type array
 * @access protected
 */
protected $all_recipients = array();

/**
 * The array of attachments.
 * @type array
 * @access protected
 */
protected $attachment = array();

/**
 * The array of custom headers.
 * @type array
 * @access protected
 */
protected $CustomHeader = array();

/**
 * The most recent Message-ID (including angular brackets).
 * @type string
 * @access protected
 */
protected $lastMessageID = '';

/**
```

```
 * The message's MIME type.
 * @type string
 * @access protected
 */
protected $message_type = '';

/**
 * The array of MIME boundary strings.
 * @type array
 * @access protected
 */
protected $boundary = array();

/**
 * The array of available languages.
 * @type array
 * @access protected
 */
protected $language = array();

/**
 * The number of errors encountered.
 * @type integer
 * @access protected
 */
protected $error_count = 0;

/**
 * The S/MIME certificate file path.
 * @type string
 * @access protected
 */
protected $sign_cert_file = '';

/**
 * The S/MIME key file path.
 * @type string
 * @access protected
 */
protected $sign_key_file = '';

/**
```

```
 * The S/MIME password for the key.
 * Used only if the key is encrypted.
 * @type string
 * @access protected
 */
protected $sign_key_pass = '';

/**
 * Whether to throw exceptions for errors.
 * @type bool
 * @access protected
 */
protected $exceptions = false;

/**
 * Error severity: message only, continue processing
 */
const STOP_MESSAGE = 0;

/**
 * Error severity: message, likely ok to continue processing
 */
const STOP_CONTINUE = 1;

/**
 * Error severity: message, plus full stop, critical error reached
 */
const STOP_CRITICAL = 2;

/**
 * SMTP RFC standard line ending
 */
const CRLF = "\r\n";

/**
 * Constructor
 * @param bool $exceptions Should we throw external exceptions?
 */
public function __construct($exceptions = false)
{
    $this->exceptions = ($exceptions == true);
    //Make sure our autoloader is loaded
    if (version_compare(PHP_VERSION, '5.1.2', '>=') and
        !spl_autoload_functions() || !in_array('PHPMailerAutoload', spl_autoload_functions())) {
        require 'PHPMailerAutoload.php';
    }
}

/**
 * Destructor.
 */
public function __destruct()
{
    if ($this->Mailer == 'smtp') { //close any open SMTP connection nicely
        $this->smtpClose();
    }
}

/**
 * Call mail() in a safe_mode-aware fashion.
 * Also, unless sendmail_path points to sendmail (or something that
 * claims to be sendmail), don't pass params (not a perfect fix,
 * but it will do)
 * @param string $to To
 * @param string $subject Subject
 * @param string $body Message Body
 * @param string $header Additional Header(s)
 * @param string $params Params
 * @access private
 * @return bool
 */
private function mailPassthru($to, $subject, $body, $header, $params)
{
    //Check overloading of mail function to avoid double-encoding
    if (ini_get('mbstring.func_overload') & 1) {
        $subject = $this->secureHeader($subject);
    } else {
        $subject = $this->encodeHeader($this->secureHeader($subject));
    }
    if (ini_get('safe_mode') || !($this->UseSendmailOptions)) {
        $rt = @mail($to, $subject, $body, $header);
    } else {
        $rt = @mail($to, $subject, $body, $header, $params);
    }
```

```
        return $rt;
    }

    /**
     * Output debugging info via user-defined method.
     * Only if debug output is enabled.
     * @see PHPMailer::$Debugoutput
     * @see PHPMailer::$SMTPDebug
     * @param string $str
     */
    protected function edebug($str)
    {
        if (!$this->SMTPDebug) {
            return;
        }
        switch ($this->Debugoutput) {
            case 'error_log':
                error_log($str);
                break;
            case 'html':
                //Cleans up output a bit for a better looking display that's HTML-safe
                echo htmlentities(preg_replace('/[\r\n]+/', '', $str), ENT_QUOTES, $this->CharSet) .
"<br>\n";
                break;
            case 'echo':
            default:
                //Just echoes exactly what was received
                echo $str;
        }
    }

    /**
     * Sets message type to HTML or plain.
     * @param bool $ishtml True for HTML mode.
     * @return void
     */
    public function isHTML($ishtml = true)
    {
        if ($ishtml) {
            $this->ContentType = 'text/html';
        } else {
            $this->ContentType = 'text/plain';
    }
}

/**
 * Send messages using SMTP.
 * @return void
 */
public function isSMTP()
{
    $this->Mailer = 'smtp';
}

/**
 * Send messages using PHP's mail() function.
 * @return void
 */
public function isMail()
{
    $this->Mailer = 'mail';
}

/**
 * Send messages using $Sendmail.
 * @return void
 */
public function isSendmail()
{
    if (!stristr(ini_get('sendmail_path'), 'sendmail')) {
        $this->Sendmail = '/usr/sbin/sendmail';
    }
    $this->Mailer = 'sendmail';
}

/**
 * Send messages using qmail.
 * @return void
 */
public function isQmail()
{
    if (!stristr(ini_get('sendmail_path'), 'qmail')) {
        $this->Sendmail = '/var/qmail/bin/qmail-inject';
    }
    $this->Mailer = 'qmail';
}

/**
 * Add a "To" address.
 * @param string $address
 * @param string $name
 * @return bool true on success, false if address already used
 */
public function addAddress($address, $name = '')
{
    return $this->addAnAddress('to', $address, $name);
}

/**
 * Add a "CC" address.
 * @note: This function works with the SMTP mailer on win32, not with the "mail" mailer.
 * @param string $address
 * @param string $name
 * @return bool true on success, false if address already used
 */
public function addCC($address, $name = '')
{
    return $this->addAnAddress('cc', $address, $name);
}

/**
 * Add a "BCC" address.
 * @note: This function works with the SMTP mailer on win32, not with the "mail" mailer.
 * @param string $address
 * @param string $name
 * @return bool true on success, false if address already used
 */
public function addBCC($address, $name = '')
{
    return $this->addAnAddress('bcc', $address, $name);
}

/**
 * Add a "Reply-to" address.
 * @param string $address
 * @param string $name
 * @return bool
 */
public function addReplyTo($address, $name = '')
{
    return $this->addAnAddress('Reply-To', $address, $name);
}

/**
 * Add an address to one of the recipient arrays.
 * Addresses that have been added already return false, but do not throw exceptions
 * @param string $kind One of 'to', 'cc', 'bcc', 'ReplyTo'
 * @param string $address The email address to send to
 * @param string $name
 * @throws phpmailerException
 * @return bool true on success, false if address already used or invalid in some way
 * @access protected
 */
protected function addAnAddress($kind, $address, $name = '')
{
    if (!preg_match('/^(to|cc|bcc|Reply-To)$/', $kind)) {
        $this->setError($this->lang('Invalid recipient array') . ': ' . $kind);
        if ($this->exceptions) {
            throw new phpmailerException('Invalid recipient array: ' . $kind);
        }
        $this->edebug($this->lang('Invalid recipient array') . ': ' . $kind);
        return false;
    }
    $address = trim($address);
    $name = trim(preg_replace('/[\r\n]+/', '', $name)); //Strip breaks and trim
    if (!$this->validateAddress($address)) {
        $this->setError($this->lang('invalid_address') . ': ' . $address);
        if ($this->exceptions) {
            throw new phpmailerException($this->lang('invalid_address') . ': ' . $address);
        }
        $this->edebug($this->lang('invalid_address') . ': ' . $address);
        return false;
    }
    if ($kind != 'Reply-To') {
        if (!isset($this->all_recipients[strtolower($address)])) {
            array_push($this->$kind, array($address, $name));
            $this->all_recipients[strtolower($address)] = true;
            return true;
        }
    } else {
        if (!array_key_exists(strtolower($address), $this->ReplyTo)) {
            $this->ReplyTo[strtolower($address)] = array($address, $name);
            return true;
        }
    }
    return false;
}

/**
 * Set the From and FromName properties.
 * @param string $address
 * @param string $name
 * @param bool $auto Whether to also set the Sender address, defaults to true
 * @throws phpmailerException
 * @return bool
 */
public function setFrom($address, $name = '', $auto = true)
{
    $address = trim($address);
    $name = trim(preg_replace('/[\r\n]+/', '', $name)); //Strip breaks and trim
    if (!$this->validateAddress($address)) {
        $this->setError($this->lang('invalid_address') . ': ' . $address);
        if ($this->exceptions) {
            throw new phpmailerException($this->lang('invalid_address') . ': ' . $address);
        }
        $this->edebug($this->lang('invalid_address') . ': ' . $address);
        return false;
    }
    $this->From = $address;
    $this->FromName = $name;
    if ($auto) {
        if (empty($this->Sender)) {
            $this->Sender = $address;
        }
    }
    return true;
}

/**
 * Return the Message-ID header of the last email.
```

```
 * Technically this is the value from the last time the headers were created,
 * but it's also the message ID of the last sent message except in
 * pathological cases.
 * @return string
 */
public function getLastMessageID()
{
    return $this->lastMessageID;
}

/**
 * Check that a string looks like an email address.
 * @param string $address The email address to check
 * @param string $patternselect A selector for the validation pattern to use :
 *   'auto' - pick best one automatically;
 *   'pcre8' - use the squiloople.com pattern, requires PCRE > 8.0, PHP >= 5.3.2, 5.2.14;
 *   'pcre' - use old PCRE implementation;
 *   'php' - use PHP built-in FILTER_VALIDATE_EMAIL; faster, less thorough;
 *   'noregex' - super fast, really dumb.
 * @return bool
 * @static
 * @access public
 */
public static function validateAddress($address, $patternselect = 'auto')
{
    if ($patternselect == 'auto') {
        if (defined(
            'PCRE_VERSION'
        )
        ) { //Check this instead of extension_loaded so it works when that function is disabled
            if (version_compare(PCRE_VERSION, '8.0') >= 0) {
                $patternselect = 'pcre8';
            } else {
                $patternselect = 'pcre';
            }
        } else {
            //Filter_var appeared in PHP 5.2.0 and does not require the PCRE extension
            if (version_compare(PHP_VERSION, '5.2.0') >= 0) {
                $patternselect = 'php';
            } else {
                $patternselect = 'noregex';
            }
        }
    }
    switch ($patternselect) {
        case 'pcre8':
            /**
             * Conforms to RFC5322: Uses *correct* regex on which FILTER_VALIDATE_EMAIL is
             * based; So why not use FILTER_VALIDATE_EMAIL? Because it was broken to
             * not allow a@b type valid addresses :(
             * @link http://squiloople.com/2009/12/20/email-address-validation/
             * @copyright 2009-2010 Michael Rushton
             * Feel free to use and redistribute this code. But please keep this copyright notice.
             */
            return (bool)preg_match(
                '/^(?!(?>(?1)"?(?>\\\[ -~]|[^"])"?(?1)){255,})(?!(?>(?1)"?(?>\\\[ -~]|[^"])"?(?1)){65,}@)' .
                '((?>(?>(?>((?>(?>(?>\x0D\x0A)?[\t ])+|(?>[\t ]*\x0D\x0A)?[\t ]+)?)(\((?>(?2)' .
                '(?>[\x01-\x08\x0B\x0C\x0E-\'*-\[\]-\x7F]|\\\[\x00-\x7F]|(?3)))*(?2)\)))+(?2))|(?2))?)' .
                '([!#-\'*+\/-9=?^-~-]+|"(?>(?2)(?>[\x01-\x08\x0B\x0C\x0E-!#-\[\]-\x7F]|\\\[\x00-\x7F]))*' .
                '(?2)")(?>(?1)\.(?1)(?4))*(?1)@(?!(?1)[a-z0-9-]{64,})(?1)(?>([a-z0-9](?>[a-z0-9-]*[a-z0-
9])?)' .
                '(?>(?1)\.(?!(?1)[a-z0-9-]{64,})(?1)(?5)){0,126}|\[(?:(?>IPv6:(?>([a-f0-
9]{1,4})(?>:(?6)){7}' .
                '|(?!(?:.*[a-f0-9][:\]]){8,})((?6)(?>:(?6)){0,6})?::(?7)?))|(?>(?>IPv6:(?>(?6)(?>:(?6)){5}:' .
                '|(?!(?:.*[a-f0-9]:){6,})(?8)?::(?>((?6)(?>:(?6)){0,4}):)?))?(25[0-5]|2[0-4][0-9]|1[0-9]{2}' .
                '|[1-9]?[0-9])(?>\.(?9)){3}))\])(?1)$/isD',
                $address
            );
            break;
        case 'pcre':
            //An older regex that doesn't need a recent PCRE
            return (bool)preg_match(
                '/^(?!(?>"?(?>\\\[ -~]|[^"])"?){255,})(?!(?>"?(?>\\\[ -~]|[^"])"?){65,}@)(?>' .
                '[!#-\'*+\/-9=?^-~-]+|"(?>(?>[\x01-\x08\x0B\x0C\x0E-!#-\[\]-\x7F]|\\\[\x00-\xFF]))*")' .
                '(?>\.(?>[!#-\'*+\/-9=?^-~-]+|"(?>(?>[\x01-\x08\x0B\x0C\x0E-!#-\[\]-\x7F]|\\\[\x00-
\xFF]))*"))*' .
                '@(?>(?![a-z0-9-]{64,})(?>[a-z0-9](?>[a-z0-9-]*[a-z0-9])?)(?>\.(?![a-z0-9-]{64,})' .
                '(?>[a-z0-9](?>[a-z0-9-]*[a-z0-9])?)){0,126}|\[(?:(?>IPv6:(?>(?>[a-f0-9]{1,4})(?>:' .
                '[a-f0-9]{1,4}){7}|(?!(?:.*[a-f0-9][:\]]){8,})(?>[a-f0-9]{1,4}(?>:[a-f0-9]{1,4}){0,6})?' .
                '::(?>[a-f0-9]{1,4}(?>:[a-f0-9]{1,4}){0,6})?))|(?>(?>IPv6:(?>[a-f0-9]{1,4}(?>:' .
                '[a-f0-9]{1,4}){5}:|(?!(?:.*[a-f0-9]:){6,})(?>[a-f0-9]{1,4}(?>:[a-f0-9]{1,4}){0,4})?' .
                '::(?>(?:[a-f0-9]{1,4}(?>:[a-f0-9]{1,4}){0,4}):)?))?(?>25[0-5]|2[0-4][0-9]|1[0-9]{2}' .
                '|[1-9]?[0-9])(?>\.(?>25[0-5]|2[0-4][0-9]|1[0-9]{2}|[1-9]?[0-9])){3}))\])$/isD',
                $address
                );
                break;
            case 'php':
            default:
                return (bool)filter_var($address, FILTER_VALIDATE_EMAIL);
                break;
            case 'noregex':
                //No PCRE! Do something _very_ approximate!
                //Check the address is 3 chars or longer and contains an @ that's not the first or last
char
                return (strlen($address) >= 3
                    and strpos($address, '@') >= 1
                    and strpos($address, '@') != strlen($address) - 1);
                break;
        }
    }

    /**
     * Create a message and send it.
     * Uses the sending method specified by $Mailer.
     * @throws phpmailerException
     * @return bool false on error - See the ErrorInfo property for details of the error.
     */
    public function send()
    {
        try {
            if (!$this->preSend()) {
                return false;
            }
            return $this->postSend();
        } catch (phpmailerException $e) {
            $this->mailHeader = '';
            $this->setError($e->getMessage());
            if ($this->exceptions) {
                throw $e;
            }
            return false;
        }
    }

    /**
     * Prepare a message for sending.
```

```
 * @throws phpmailerException
 * @return bool
 */
public function preSend()
{
    try {
        $this->mailHeader = "";
        if ((count($this->to) + count($this->cc) + count($this->bcc)) < 1) {
            throw new phpmailerException($this->lang('provide_address'), self::STOP_CRITICAL);
        }

        // Set whether the message is multipart/alternative
        if (!empty($this->AltBody)) {
            $this->ContentType = 'multipart/alternative';
        }

        $this->error_count = 0; // reset errors
        $this->setMessageType();
        // Refuse to send an empty message unless we are specifically allowing it
        if (!$this->AllowEmpty and empty($this->Body)) {
            throw new phpmailerException($this->lang('empty_message'), self::STOP_CRITICAL);
        }

        $this->MIMEHeader = $this->createHeader();
        $this->MIMEBody = $this->createBody();

        // To capture the complete message when using mail(), create
        // an extra header list which createHeader() doesn't fold in
        if ($this->Mailer == 'mail') {
            if (count($this->to) > 0) {
                $this->mailHeader .= $this->addrAppend("To", $this->to);
            } else {
                $this->mailHeader .= $this->headerLine("To", "undisclosed-recipients:;");
            }
            $this->mailHeader .= $this->headerLine(
                'Subject',
                $this->encodeHeader($this->secureHeader(trim($this->Subject)))
            );
        }

        // Sign with DKIM if enabled
        if (!empty($this->DKIM_domain)
            && !empty($this->DKIM_private)
            && !empty($this->DKIM_selector)
            && !empty($this->DKIM_domain)
            && file_exists($this->DKIM_private)) {
            $header_dkim = $this->DKIM_Add(
                $this->MIMEHeader . $this->mailHeader,
                $this->encodeHeader($this->secureHeader($this->Subject)),
                $this->MIMEBody
            );
            $this->MIMEHeader = rtrim($this->MIMEHeader, "\r\n ") . self::CRLF .
                str_replace("\r\n", "\n", $header_dkim) . self::CRLF;
        }
        return true;

    } catch (phpmailerException $e) {
        $this->setError($e->getMessage());
        if ($this->exceptions) {
            throw $e;
        }
        return false;
    }
}

/**
 * Actually send a message.
 * Send the email via the selected mechanism
 * @throws phpmailerException
 * @return bool
 */
public function postSend()
{
    try {
        // Choose the mailer and send through it
        switch ($this->Mailer) {
            case 'sendmail':
            case 'qmail':
                return $this->sendmailSend($this->MIMEHeader, $this->MIMEBody);
            case 'smtp':
                return $this->smtpSend($this->MIMEHeader, $this->MIMEBody);
            case 'mail':
                return $this->mailSend($this->MIMEHeader, $this->MIMEBody);
            default:
                if (method_exists($this, $this->Mailer.'Send')) {
                    $sendMethod = $this->Mailer.'Send';
                    return $this->$sendMethod($this->MIMEHeader, $this->MIMEBody);
                } else {
                    return $this->mailSend($this->MIMEHeader, $this->MIMEBody);
                }
            }
        } catch (phpmailerException $e) {
            $this->setError($e->getMessage());
            if ($this->exceptions) {
                throw $e;
            }
            $this->edebug($e->getMessage() . "\n");
        }
        return false;
    }

    /**
     * Send mail using the $Sendmail program.
     * @param string $header The message headers
     * @param string $body The message body
     * @see PHPMailer::$Sendmail
     * @throws phpmailerException
     * @access protected
     * @return bool
     */
    protected function sendmailSend($header, $body)
    {
        if ($this->Sender != '') {
            if ($this->Mailer == 'qmail') {
                $sendmail = sprintf("%s -f%s", escapeshellcmd($this->Sendmail), escapeshellarg($this-
>Sender));
            } else {
                $sendmail = sprintf("%s -oi -f%s -t", escapeshellcmd($this->Sendmail),
escapeshellarg($this->Sender));
            }
        } else {
            if ($this->Mailer == 'qmail') {
                $sendmail = sprintf("%s", escapeshellcmd($this->Sendmail));
            } else {
                $sendmail = sprintf("%s -oi -t", escapeshellcmd($this->Sendmail));
            }
        }
        if ($this->SingleTo === true) {
            foreach ($this->SingleToArray as $val) {
                if (!@$mail = popen($sendmail, 'w')) {
                    throw new phpmailerException($this->lang('execute') . $this->Sendmail,
self::STOP_CRITICAL);
                }
                fputs($mail, "To: " . $val . "\n");
                fputs($mail, $header);
                fputs($mail, $body);
                $result = pclose($mail);
                // implement call back function if it exists
                $isSent = ($result == 0) ? 1 : 0;
                $this->doCallback($isSent, $val, $this->cc, $this->bcc, $this->Subject, $body, $this-
>From);
                if ($result != 0) {
                    throw new phpmailerException($this->lang('execute') . $this->Sendmail,
self::STOP_CRITICAL);
                }
            }
        } else {
            if (!@$mail = popen($sendmail, 'w')) {
                throw new phpmailerException($this->lang('execute') . $this->Sendmail,
self::STOP_CRITICAL);
            }
            fputs($mail, $header);
            fputs($mail, $body);
            $result = pclose($mail);
            // implement call back function if it exists
            $isSent = ($result == 0) ? 1 : 0;
            $this->doCallback($isSent, $this->to, $this->cc, $this->bcc, $this->Subject, $body, $this-
>From);
            if ($result != 0) {
                throw new phpmailerException($this->lang('execute') . $this->Sendmail,
self::STOP_CRITICAL);
            }
        }
        return true;
    }

    /**
     * Send mail using the PHP mail() function.
     * @param string $header The message headers
     * @param string $body The message body
     * @link http://www.php.net/manual/en/book.mail.php
     * @throws phpmailerException
     * @access protected
     * @return bool
     */
    protected function mailSend($header, $body)
    {
        $toArr = array();
        foreach ($this->to as $t) {
            $toArr[] = $this->addrFormat($t);
        }
        $to = implode(', ', $toArr);

        if (empty($this->Sender)) {
            $params = " ";
        } else {
            $params = sprintf("-f%s", $this->Sender);
        }
        if ($this->Sender != '' and !ini_get('safe_mode')) {
            $old_from = ini_get('sendmail_from');
            ini_set('sendmail_from', $this->Sender);
        }
        $rt = false;
        if ($this->SingleTo === true && count($toArr) > 1) {
            foreach ($toArr as $val) {
                $rt = $this->mailPassthru($val, $this->Subject, $body, $header, $params);
                // implement call back function if it exists
                $isSent = ($rt == 1) ? 1 : 0;
                $this->doCallback($isSent, $val, $this->cc, $this->bcc, $this->Subject, $body, $this-
>From);
            }
        } else {
            $rt = $this->mailPassthru($to, $this->Subject, $body, $header, $params);
            // implement call back function if it exists
            $isSent = ($rt == 1) ? 1 : 0;
            $this->doCallback($isSent, $to, $this->cc, $this->bcc, $this->Subject, $body, $this->From);
        }
        if (isset($old_from)) {
            ini_set('sendmail_from', $old_from);
        }
        if (!$rt) {
            throw new phpmailerException($this->lang('instantiate'), self::STOP_CRITICAL);
        }
        return true;
    }

    /**
     * Get an instance to use for SMTP operations.
     * Override this function to load your own SMTP implementation
     * @return SMTP
     */
    public function getSMTPInstance()
    {
        if (!is_object($this->smtp)) {
            $this->smtp = new SMTP;
        }
        return $this->smtp;
    }

    /**
     * Send mail via SMTP.
     * Returns false if there is a bad MAIL FROM, RCPT, or DATA input.
     * Uses the PHPMailerSMTP class by default.
     * @see PHPMailer::getSMTPInstance() to use a different class.
     * @param string $header The message headers
     * @param string $body The message body
     * @throws phpmailerException
     * @uses SMTP
     * @access protected
     * @return bool
     */
    protected function smtpSend($header, $body)
    {
        $bad_rcpt = array();

        if (!$this->smtpConnect()) {
            throw new phpmailerException($this->lang('smtp_connect_failed'), self::STOP_CRITICAL);
        }
        $smtp_from = ($this->Sender == '') ? $this->From : $this->Sender;
        if (!$this->smtp->mail($smtp_from)) {
            $this->setError($this->lang('from_failed') . $smtp_from . ' : ' . implode(',', $this->smtp-
>getError()));
```

```
    throw new phpmailerException($this->ErrorInfo, self::STOP_CRITICAL);
}

// Attempt to send attach all recipients
foreach ($this->to as $to) {
    if (!$this->smtp->recipient($to[0])) {
        $bad_rcpt[] = $to[0];
        $isSent = 0;
    } else {
        $isSent = 1;
    }
    $this->doCallback($isSent, $to[0], '', '', $this->Subject, $body, $this->From);
}
foreach ($this->cc as $cc) {
    if (!$this->smtp->recipient($cc[0])) {
        $bad_rcpt[] = $cc[0];
        $isSent = 0;
    } else {
        $isSent = 1;
    }
    $this->doCallback($isSent, '', $cc[0], '', $this->Subject, $body, $this->From);
}
foreach ($this->bcc as $bcc) {
    if (!$this->smtp->recipient($bcc[0])) {
        $bad_rcpt[] = $bcc[0];
        $isSent = 0;
    } else {
        $isSent = 1;
    }
    $this->doCallback($isSent, '', '', $bcc[0], $this->Subject, $body, $this->From);
}

if (!$this->smtp->data($header . $body)) {
    throw new phpmailerException($this->lang('data_not_accepted'), self::STOP_CRITICAL);
}
if ($this->SMTPKeepAlive == true) {
    $this->smtp->reset();
} else {
    $this->smtp->quit();
    $this->smtp->close();
}
if (count($bad_rcpt) > 0) { //Create error message for any bad addresses
```

```
            throw new phpmailerException(
                $this->lang('recipients_failed') . implode(', ', $bad_rcpt),
                self::STOP_CONTINUE
            );
        }
        return true;
    }

    /**
     * Initiate a connection to an SMTP server.
     * Returns false if the operation failed.
     * @param array $options An array of options compatible with stream_context_create()
     * @uses SMTP
     * @access public
     * @throws phpmailerException
     * @return bool
     */
    public function smtpConnect($options = array())
    {
        if (is_null($this->smtp)) {
            $this->smtp = $this->getSMTPInstance();
        }

        //Already connected?
        if ($this->smtp->connected()) {
            return true;
        }

        $this->smtp->setTimeout($this->Timeout);
        $this->smtp->setDebugLevel($this->SMTPDebug);
        $this->smtp->setDebugOutput($this->Debugoutput);
        $this->smtp->setVerp($this->do_verp);
        $hosts = explode(';', $this->Host);
        $lastexception = null;

        foreach ($hosts as $hostentry) {
            $hostinfo = array();
            if (!preg_match('/^((ssl|tls):\/\/)*([a-zA-Z0-9\.-]*):?([0-9]*)$/', trim($hostentry), $hostinfo)) {
                //Not a valid host entry
                continue;
            }
            //$hostinfo[2]: optional ssl or tls prefix
```

```
//$hostinfo[3]: the hostname
//$hostinfo[4]: optional port number
//The host string prefix can temporarily override the current setting for SMTPSecure
//If it's not specified, the default value is used
$prefix = '';
$tls = ($this->SMTPSecure == 'tls');
if ($hostinfo[2] == 'ssl' or ($hostinfo[2] == '' and $this->SMTPSecure == 'ssl')) {
    $prefix = 'ssl://';
    $tls = false; //Can't have SSL and TLS at once
} elseif ($hostinfo[2] == 'tls') {
    $tls = true;
    //tls doesn't use a prefix
}
$host = $hostinfo[3];
$port = $this->Port;
$tport = (integer)$hostinfo[4];
if ($tport > 0 and $tport < 65536) {
    $port = $tport;
}
if ($this->smtp->connect($prefix . $host, $port, $this->Timeout, $options)) {
    try {
        if ($this->Helo) {
            $hello = $this->Helo;
        } else {
            $hello = $this->serverHostname();
        }
        $this->smtp->hello($hello);

        if ($tls) {
            if (!$this->smtp->startTLS()) {
                throw new phpmailerException($this->lang('connect_host'));
            }
            //We must resend HELO after tls negotiation
            $this->smtp->hello($hello);
        }
        if ($this->SMTPAuth) {
            if (!$this->smtp->authenticate(
                $this->Username,
                $this->Password,
                $this->AuthType,
                $this->Realm,
                $this->Workstation
```

```
                    )
                    ) {
                        throw new phpmailerException($this->lang('authenticate'));
                    }
                }
                return true;
            } catch (phpmailerException $e) {
                $lastexception = $e;
                //We must have connected, but then failed TLS or Auth, so close connection nicely
                $this->smtp->quit();
            }
        }
    }
    //If we get here, all connection attempts have failed, so close connection hard
    $this->smtp->close();
    //As we've caught all exceptions, just report whatever the last one was
    if ($this->exceptions and !is_null($lastexception)) {
        throw $lastexception;
    }
    return false;
}

/**
 * Close the active SMTP session if one exists.
 * @return void
 */
public function smtpClose()
{
    if ($this->smtp !== null) {
        if ($this->smtp->connected()) {
            $this->smtp->quit();
            $this->smtp->close();
        }
    }
}

/**
 * Set the language for error messages.
 * Returns false if it cannot load the language file.
 * The default language is English.
 * @param string $langcode ISO 639-1 2-character language code (e.g. French is "fr")
 * @param string $lang_path Path to the language file directory, with trailing separator (slash)
 * @return bool
 * @access public
 */
public function setLanguage($langcode = 'en', $lang_path = 'language/')
{
    //Define full set of translatable strings
    $PHPMAILER_LANG = array(
        'authenticate' => 'SMTP Error: Could not authenticate.',
        'connect_host' => 'SMTP Error: Could not connect to SMTP host.',
        'data_not_accepted' => 'SMTP Error: data not accepted.',
        'empty_message' => 'Message body empty',
        'encoding' => 'Unknown encoding: ',
        'execute' => 'Could not execute: ',
        'file_access' => 'Could not access file: ',
        'file_open' => 'File Error: Could not open file: ',
        'from_failed' => 'The following From address failed: ',
        'instantiate' => 'Could not instantiate mail function.',
        'invalid_address' => 'Invalid address',
        'mailer_not_supported' => ' mailer is not supported.',
        'provide_address' => 'You must provide at least one recipient email address.',
        'recipients_failed' => 'SMTP Error: The following recipients failed: ',
        'signing' => 'Signing Error: ',
        'smtp_connect_failed' => 'SMTP connect() failed.',
        'smtp_error' => 'SMTP server error: ',
        'variable_set' => 'Cannot set or reset variable: '
    );
    //Overwrite language-specific strings.
    //This way we'll never have missing translations - no more "language string failed to load"!
    $l = true;
    if ($langcode != 'en') { //There is no English translation file
        $l = @include $lang_path . 'phpmailer.lang-' . $langcode . '.php';
    }
    $this->language = $PHPMAILER_LANG;
    return ($l == true); //Returns false if language not found
}

/**
 * Get the array of strings for the current language.
 * @return array
 */
public function getTranslations()
{
        return $this->language;
    }

    /**
     * Create recipient headers.
     * @access public
     * @param string $type
     * @param array $addr An array of recipient,
     * where each recipient is a 2-element indexed array with element 0 containing an address
     * and element 1 containing a name, like:
     * array(array('joe@example.com', 'Joe User'), array('zoe@example.com', 'Zoe User'))
     * @return string
     */
    public function addrAppend($type, $addr)
    {
        $addresses = array();
        foreach ($addr as $a) {
            $addresses[] = $this->addrFormat($a);
        }
        return $type . ': ' . implode(', ', $addresses) . $this->LE;
    }

    /**
     * Format an address for use in a message header.
     * @access public
     * @param array $addr A 2-element indexed array, element 0 containing an address, element 1
containing a name
     *      like array('joe@example.com', 'Joe User')
     * @return string
     */
    public function addrFormat($addr)
    {
        if (empty($addr[1])) { // No name provided
            return $this->secureHeader($addr[0]);
        } else {
            return $this->encodeHeader($this->secureHeader($addr[1]), 'phrase') . " <" . $this-
>secureHeader(
                $addr[0]
            ) . ">";
        }
    }
```

```
/**
 * Word-wrap message.
 * For use with mailers that do not automatically perform wrapping
 * and for quoted-printable encoded messages.
 * Original written by philippe.
 * @param string $message The message to wrap
 * @param integer $length The line length to wrap to
 * @param bool $qp_mode Whether to run in Quoted-Printable mode
 * @access public
 * @return string
 */
public function wrapText($message, $length, $qp_mode = false)
{
    $soft_break = ($qp_mode) ? sprintf(" =%s", $this->LE) : $this->LE;
    // If utf-8 encoding is used, we will need to make sure we don't
    // split multibyte characters when we wrap
    $is_utf8 = (strtolower($this->CharSet) == "utf-8");
    $lelen = strlen($this->LE);
    $crlflen = strlen(self::CRLF);

    $message = $this->fixEOL($message);
    if (substr($message, -$lelen) == $this->LE) {
        $message = substr($message, 0, -$lelen);
    }

    $line = explode($this->LE, $message); // Magic. We know fixEOL uses $LE
    $message = '';
    for ($i = 0; $i < count($line); $i++) {
        $line_part = explode(' ', $line[$i]);
        $buf = '';
        for ($e = 0; $e < count($line_part); $e++) {
            $word = $line_part[$e];
            if ($qp_mode and (strlen($word) > $length)) {
                $space_left = $length - strlen($buf) - $crlflen;
                if ($e != 0) {
                    if ($space_left > 20) {
                        $len = $space_left;
                        if ($is_utf8) {
                            $len = $this->utf8CharBoundary($word, $len);
                        } elseif (substr($word, $len - 1, 1) == "=") {
                            $len--;
                        } elseif (substr($word, $len - 2, 1) == "=") {
```

```
                    $len -= 2;
                }
                $part = substr($word, 0, $len);
                $word = substr($word, $len);
                $buf .= ' ' . $part;
                $message .= $buf . sprintf("=%s", self::CRLF);
            } else {
                $message .= $buf . $soft_break;
            }
            $buf = '';
        }
        while (strlen($word) > 0) {
            if ($length <= 0) {
                break;
            }
            $len = $length;
            if ($is_utf8) {
                $len = $this->utf8CharBoundary($word, $len);
            } elseif (substr($word, $len - 1, 1) == "=") {
                $len--;
            } elseif (substr($word, $len - 2, 1) == "=") {
                $len -= 2;
            }
            $part = substr($word, 0, $len);
            $word = substr($word, $len);

            if (strlen($word) > 0) {
                $message .= $part . sprintf("=%s", self::CRLF);
            } else {
                $buf = $part;
            }
        }
    } else {
        $buf_o = $buf;
        $buf .= ($e == 0) ? $word : (' ' . $word);

        if (strlen($buf) > $length and $buf_o != '') {
            $message .= $buf_o . $soft_break;
            $buf = $word;
        }
    }
}
```

```
            $message .= $buf . self::CRLF;
        }

        return $message;
    }

    /**
     * Find the last character boundary prior to $maxLength in a utf-8
     * quoted (printable) encoded string.
     * Original written by Colin Brown.
     * @access public
     * @param string $encodedText utf-8 QP text
     * @param int $maxLength   find last character boundary prior to this length
     * @return int
     */
    public function utf8CharBoundary($encodedText, $maxLength)
    {
        $foundSplitPos = false;
        $lookBack = 3;
        while (!$foundSplitPos) {
            $lastChunk = substr($encodedText, $maxLength - $lookBack, $lookBack);
            $encodedCharPos = strpos($lastChunk, "=");
            if ($encodedCharPos !== false) {
                // Found start of encoded character byte within $lookBack block.
                // Check the encoded byte value (the 2 chars after the '=')
                $hex = substr($encodedText, $maxLength - $lookBack + $encodedCharPos + 1, 2);
                $dec = hexdec($hex);
                if ($dec < 128) { // Single byte character.
                    // If the encoded char was found at pos 0, it will fit
                    // otherwise reduce maxLength to start of the encoded char
                    $maxLength = ($encodedCharPos == 0) ? $maxLength :
                        $maxLength - ($lookBack - $encodedCharPos);
                    $foundSplitPos = true;
                } elseif ($dec >= 192) { // First byte of a multi byte character
                    // Reduce maxLength to split at start of character
                    $maxLength = $maxLength - ($lookBack - $encodedCharPos);
                    $foundSplitPos = true;
                } elseif ($dec < 192) { // Middle byte of a multi byte character, look further back
                    $lookBack += 3;
                }
            } else {
                // No encoded character found
                $foundSplitPos = true;
            }
        }
        return $maxLength;
    }


    /**
     * Set the body wrapping.
     * @access public
     * @return void
     */
    public function setWordWrap()
    {
        if ($this->WordWrap < 1) {
            return;
        }

        switch ($this->message_type) {
            case 'alt':
            case 'alt_inline':
            case 'alt_attach':
            case 'alt_inline_attach':
                $this->AltBody = $this->wrapText($this->AltBody, $this->WordWrap);
                break;
            default:
                $this->Body = $this->wrapText($this->Body, $this->WordWrap);
                break;
        }
    }

    /**
     * Assemble message headers.
     * @access public
     * @return string The assembled headers
     */
    public function createHeader()
    {
        $result = '';

        // Set the boundaries
        $uniq_id = md5(uniqid(time()));
```

```
$this->boundary[1] = 'b1_' . $uniq_id;
$this->boundary[2] = 'b2_' . $uniq_id;
$this->boundary[3] = 'b3_' . $uniq_id;

if ($this->MessageDate == '') {
    $result .= $this->headerLine('Date', self::rfcDate());
} else {
    $result .= $this->headerLine('Date', $this->MessageDate);
}

if ($this->ReturnPath) {
    $result .= $this->headerLine('Return-Path', '<' . trim($this->ReturnPath) . '>');
} elseif ($this->Sender == '') {
    $result .= $this->headerLine('Return-Path', '<' . trim($this->From) . '>');
} else {
    $result .= $this->headerLine('Return-Path', '<' . trim($this->Sender) . '>');
}

// To be created automatically by mail()
if ($this->Mailer != 'mail') {
    if ($this->SingleTo === true) {
        foreach ($this->to as $t) {
            $this->SingleToArray[] = $this->addrFormat($t);
        }
    } else {
        if (count($this->to) > 0) {
            $result .= $this->addrAppend('To', $this->to);
        } elseif (count($this->cc) == 0) {
            $result .= $this->headerLine('To', 'undisclosed-recipients:;');
        }
    }
}

$result .= $this->addrAppend('From', array(array(trim($this->From), $this->FromName)));

// sendmail and mail() extract Cc from the header before sending
if (count($this->cc) > 0) {
    $result .= $this->addrAppend('Cc', $this->cc);
}

// sendmail and mail() extract Bcc from the header before sending
if ((
                $this->Mailer == 'sendmail' or $this->Mailer == 'qmail' or $this->Mailer == 'mail'
            )
            and count($this->bcc) > 0
        ) {
            $result .= $this->addrAppend('Bcc', $this->bcc);
        }

        if (count($this->ReplyTo) > 0) {
            $result .= $this->addrAppend('Reply-To', $this->ReplyTo);
        }

        // mail() sets the subject itself
        if ($this->Mailer != 'mail') {
            $result .= $this->headerLine('Subject', $this->encodeHeader($this->secureHeader($this-
>Subject)));
        }

        if ($this->MessageID != '') {
            $this->lastMessageID = $this->MessageID;
        } else {
            $this->lastMessageID = sprintf("<%s@%s>", $uniq_id, $this->ServerHostname());
        }
        $result .= $this->HeaderLine('Message-ID', $this->lastMessageID);
        $result .= $this->headerLine('X-Priority', $this->Priority);
        if ($this->XMailer == '') {
            $result .= $this->headerLine(
                'X-Mailer',
                'PHPMailer ' . $this->Version . ' (https://github.com/PHPMailer/PHPMailer/)'
            );
        } else {
            $myXmailer = trim($this->XMailer);
            if ($myXmailer) {
                $result .= $this->headerLine('X-Mailer', $myXmailer);
            }
        }

        if ($this->ConfirmReadingTo != '') {
            $result .= $this->headerLine('Disposition-Notification-To', '<' . trim($this-
>ConfirmReadingTo) . '>');
        }

        // Add custom headers
```

```
    for ($index = 0; $index < count($this->CustomHeader); $index++) {
        $result .= $this->headerLine(
            trim($this->CustomHeader[$index][0]),
            $this->encodeHeader(trim($this->CustomHeader[$index][1]))
        );
    }
    if (!$this->sign_key_file) {
        $result .= $this->headerLine('MIME-Version', '1.0');
        $result .= $this->getMailMIME();
    }

    return $result;
}

/**
 * Get the message MIME type headers.
 * @access public
 * @return string
 */
public function getMailMIME()
{
    $result = '';
    switch ($this->message_type) {
        case 'inline':
            $result .= $this->headerLine('Content-Type', 'multipart/related;');
            $result .= $this->textLine("\tboundary=\"" . $this->boundary[1] . '"');
            break;
        case 'attach':
        case 'inline_attach':
        case 'alt_attach':
        case 'alt_inline_attach':
            $result .= $this->headerLine('Content-Type', 'multipart/mixed;');
            $result .= $this->textLine("\tboundary=\"" . $this->boundary[1] . '"');
            break;
        case 'alt':
        case 'alt_inline':
            $result .= $this->headerLine('Content-Type', 'multipart/alternative;');
            $result .= $this->textLine("\tboundary=\"" . $this->boundary[1] . '"');
            break;
        default:
            // Catches case 'plain': and case '':
            $result .= $this->textLine('Content-Type: ' . $this->ContentType . '; charset=' . $this-
>CharSet);
                break;
        }
        //RFC1341 part 5 says 7bit is assumed if not specified
        if ($this->Encoding != '7bit') {
            $result .= $this->headerLine('Content-Transfer-Encoding', $this->Encoding);
        }

        if ($this->Mailer != 'mail') {
            $result .= $this->LE;
        }

        return $result;
    }

    /**
     * Returns the whole MIME message.
     * Includes complete headers and body.
     * Only valid post PreSend().
     * @see PHPMailer::PreSend()
     * @access public
     * @return string
     */
    public function getSentMIMEMessage()
    {
        return $this->MIMEHeader . $this->mailHeader . self::CRLF . $this->MIMEBody;
    }


    /**
     * Assemble the message body.
     * Returns an empty string on failure.
     * @access public
     * @throws phpmailerException
     * @return string The assembled message body
     */
    public function createBody()
    {
        $body = '';

        if ($this->sign_key_file) {
            $body .= $this->getMailMIME() . $this->LE;
        }

        $this->setWordWrap();

        switch ($this->message_type) {
            case 'inline':
                $body .= $this->getBoundary($this->boundary[1], '', '', '');
                $body .= $this->encodeString($this->Body, $this->Encoding);
                $body .= $this->LE . $this->LE;
                $body .= $this->attachAll('inline', $this->boundary[1]);
                break;
            case 'attach':
                $body .= $this->getBoundary($this->boundary[1], '', '', '');
                $body .= $this->encodeString($this->Body, $this->Encoding);
                $body .= $this->LE . $this->LE;
                $body .= $this->attachAll('attachment', $this->boundary[1]);
                break;
            case 'inline_attach':
                $body .= $this->textLine('--' . $this->boundary[1]);
                $body .= $this->headerLine('Content-Type', 'multipart/related;');
                $body .= $this->textLine("\tboundary=\"" . $this->boundary[2] . '"');
                $body .= $this->LE;
                $body .= $this->getBoundary($this->boundary[2], '', '', '');
                $body .= $this->encodeString($this->Body, $this->Encoding);
                $body .= $this->LE . $this->LE;
                $body .= $this->attachAll('inline', $this->boundary[2]);
                $body .= $this->LE;
                $body .= $this->attachAll('attachment', $this->boundary[1]);
                break;
            case 'alt':
                $body .= $this->getBoundary($this->boundary[1], '', 'text/plain', '');
                $body .= $this->encodeString($this->AltBody, $this->Encoding);
                $body .= $this->LE . $this->LE;
                $body .= $this->getBoundary($this->boundary[1], '', 'text/html', '');
                $body .= $this->encodeString($this->Body, $this->Encoding);
                $body .= $this->LE . $this->LE;
                if (!empty($this->Ical)) {
                    $body .= $this->getBoundary($this->boundary[1], '', 'text/calendar;
method=REQUEST', '');
                    $body .= $this->encodeString($this->Ical, $this->Encoding);
                    $body .= $this->LE . $this->LE;
                }
                $body .= $this->endBoundary($this->boundary[1]);
                break;
            case 'alt_inline':
                $body .= $this->getBoundary($this->boundary[1], '', 'text/plain', '');
                $body .= $this->encodeString($this->AltBody, $this->Encoding);
                $body .= $this->LE . $this->LE;
                $body .= $this->textLine('--' . $this->boundary[1]);
                $body .= $this->headerLine('Content-Type', 'multipart/related;');
                $body .= $this->textLine("\tboundary=\"" . $this->boundary[2] . '"');
                $body .= $this->LE;
                $body .= $this->getBoundary($this->boundary[2], '', 'text/html', '');
                $body .= $this->encodeString($this->Body, $this->Encoding);
                $body .= $this->LE . $this->LE;
                $body .= $this->attachAll('inline', $this->boundary[2]);
                $body .= $this->LE;
                $body .= $this->endBoundary($this->boundary[1]);
                break;
            case 'alt_attach':
                $body .= $this->textLine('--' . $this->boundary[1]);
                $body .= $this->headerLine('Content-Type', 'multipart/alternative;');
                $body .= $this->textLine("\tboundary=\"" . $this->boundary[2] . '"');
                $body .= $this->LE;
                $body .= $this->getBoundary($this->boundary[2], '', 'text/plain', '');
                $body .= $this->encodeString($this->AltBody, $this->Encoding);
                $body .= $this->LE . $this->LE;
                $body .= $this->getBoundary($this->boundary[2], '', 'text/html', '');
                $body .= $this->encodeString($this->Body, $this->Encoding);
                $body .= $this->LE . $this->LE;
                $body .= $this->endBoundary($this->boundary[2]);
                $body .= $this->LE;
                $body .= $this->attachAll('attachment', $this->boundary[1]);
                break;
            case 'alt_inline_attach':
                $body .= $this->textLine('--' . $this->boundary[1]);
                $body .= $this->headerLine('Content-Type', 'multipart/alternative;');
                $body .= $this->textLine("\tboundary=\"" . $this->boundary[2] . '"');
                $body .= $this->LE;
                $body .= $this->getBoundary($this->boundary[2], '', 'text/plain', '');
                $body .= $this->encodeString($this->AltBody, $this->Encoding);
                $body .= $this->LE . $this->LE;
                $body .= $this->textLine('--' . $this->boundary[2]);
                $body .= $this->headerLine('Content-Type', 'multipart/related;');
```

```
                $body .= $this->textLine("\tboundary=\"" . $this->boundary[3] . '"');
                $body .= $this->LE;
                $body .= $this->getBoundary($this->boundary[3], '', 'text/html', '');
                $body .= $this->encodeString($this->Body, $this->Encoding);
                $body .= $this->LE . $this->LE;
                $body .= $this->attachAll('inline', $this->boundary[3]);
                $body .= $this->LE;
                $body .= $this->endBoundary($this->boundary[2]);
                $body .= $this->LE;
                $body .= $this->attachAll('attachment', $this->boundary[1]);
                break;
            default:
                // catch case 'plain' and case ''
                $body .= $this->encodeString($this->Body, $this->Encoding);
                break;
        }

        if ($this->isError()) {
            $body = '';
        } elseif ($this->sign_key_file) {
            try {
                if (!defined('PKCS7_TEXT')) {
                    throw new phpmailerException($this->lang('signing') . ' OpenSSL extension
missing.');
                }
                //TODO would be nice to use php://temp streams here, but need to wrap for PHP < 5.1
                $file = tempnam(sys_get_temp_dir(), 'mail');
                file_put_contents($file, $body); //TODO check this worked
                $signed = tempnam(sys_get_temp_dir(), 'signed');
                if (@openssl_pkcs7_sign(
                    $file,
                    $signed,
                    'file://' . realpath($this->sign_cert_file),
                    array('file://' . realpath($this->sign_key_file), $this->sign_key_pass),
                    null
                )
                ) {
                    @unlink($file);
                    $body = file_get_contents($signed);
                    @unlink($signed);
                } else {
                    @unlink($file);
                @unlink($signed);
                throw new phpmailerException($this->lang('signing') . openssl_error_string());
            }
        } catch (phpmailerException $e) {
            $body = '';
            if ($this->exceptions) {
                throw $e;
            }
        }
    }
    return $body;
}

/**
 * Return the start of a message boundary.
 * @access protected
 * @param string $boundary
 * @param string $charSet
 * @param string $contentType
 * @param string $encoding
 * @return string
 */
protected function getBoundary($boundary, $charSet, $contentType, $encoding)
{
    $result = '';
    if ($charSet == '') {
        $charSet = $this->CharSet;
    }
    if ($contentType == '') {
        $contentType = $this->ContentType;
    }
    if ($encoding == '') {
        $encoding = $this->Encoding;
    }
    $result .= $this->textLine('--' . $boundary);
    $result .= sprintf("Content-Type: %s; charset=%s", $contentType, $charSet);
    $result .= $this->LE;
    $result .= $this->headerLine('Content-Transfer-Encoding', $encoding);
    $result .= $this->LE;

    return $result;
}
```

```
/**
 * Return the end of a message boundary.
 * @access protected
 * @param string $boundary
 * @return string
 */
protected function endBoundary($boundary)
{
    return $this->LE . '--' . $boundary . '--' . $this->LE;
}

/**
 * Set the message type.
 * PHPMailer only supports some preset message types,
 * not arbitrary MIME structures.
 * @access protected
 * @return void
 */
protected function setMessageType()
{
    $this->message_type = array();
    if ($this->alternativeExists()) {
        $this->message_type[] = "alt";
    }
    if ($this->inlineImageExists()) {
        $this->message_type[] = "inline";
    }
    if ($this->attachmentExists()) {
        $this->message_type[] = "attach";
    }
    $this->message_type = implode("_", $this->message_type);
    if ($this->message_type == "") {
        $this->message_type = "plain";
    }
}

/**
 * Format a header line.
 * @access public
 * @param string $name
 * @param string $value
```

```
     * @return string
     */
    public function headerLine($name, $value)
    {
        return $name . ': ' . $value . $this->LE;
    }

    /**
     * Return a formatted mail line.
     * @access public
     * @param string $value
     * @return string
     */
    public function textLine($value)
    {
        return $value . $this->LE;
    }

    /**
     * Add an attachment from a path on the filesystem.
     * Returns false if the file could not be found or read.
     * @param string $path Path to the attachment.
     * @param string $name Overrides the attachment name.
     * @param string $encoding File encoding (see $Encoding).
     * @param string $type File extension (MIME) type.
     * @param string $disposition Disposition to use
     * @throws phpmailerException
     * @return bool
     */
    public function addAttachment($path, $name = '', $encoding = 'base64', $type = '', $disposition
= 'attachment')
    {
        try {
            if (!@is_file($path)) {
                throw new phpmailerException($this->lang('file_access') . $path,
self::STOP_CONTINUE);
            }

            //If a MIME type is not specified, try to work it out from the file name
            if ($type == '') {
                $type = self::filenameToType($path);
            }
        $filename = basename($path);
        if ($name == '') {
            $name = $filename;
        }

        $this->attachment[] = array(
            0 => $path,
            1 => $filename,
            2 => $name,
            3 => $encoding,
            4 => $type,
            5 => false, // isStringAttachment
            6 => $disposition,
            7 => 0
        );

    } catch (phpmailerException $e) {
        $this->setError($e->getMessage());
        if ($this->exceptions) {
            throw $e;
        }
        $this->edebug($e->getMessage() . "\n");
        return false;
    }
    return true;
}

/**
 * Return the array of attachments.
 * @return array
 */
public function getAttachments()
{
    return $this->attachment;
}

/**
 * Attach all file, string, and binary attachments to the message.
 * Returns an empty string on failure.
 * @access protected
 * @param string $disposition_type
 * @param string $boundary
 * @return string
 */
protected function attachAll($disposition_type, $boundary)
{
    // Return text of body
    $mime = array();
    $cidUniq = array();
    $incl = array();

    // Add all attachments
    foreach ($this->attachment as $attachment) {
        // Check if it is a valid disposition_filter
        if ($attachment[6] == $disposition_type) {
            // Check for string attachment
            $string = '';
            $path = '';
            $bString = $attachment[5];
            if ($bString) {
                $string = $attachment[0];
            } else {
                $path = $attachment[0];
            }

            $inclhash = md5(serialize($attachment));
            if (in_array($inclhash, $incl)) {
                continue;
            }
            $incl[] = $inclhash;
            $name = $attachment[2];
            $encoding = $attachment[3];
            $type = $attachment[4];
            $disposition = $attachment[6];
            $cid = $attachment[7];
            if ($disposition == 'inline' && isset($cidUniq[$cid])) {
                continue;
            }
            $cidUniq[$cid] = true;

            $mime[] = sprintf("--%s%s", $boundary, $this->LE);
            $mime[] = sprintf(
                "Content-Type: %s; name=\"%s\"%s",
                $type,
                $this->encodeHeader($this->secureHeader($name)),
                $this->LE
            );
            $mime[] = sprintf("Content-Transfer-Encoding: %s%s", $encoding, $this->LE);

            if ($disposition == 'inline') {
                $mime[] = sprintf("Content-ID: <%s>%s", $cid, $this->LE);
            }

            // If a filename contains any of these chars, it should be quoted,
            // but not otherwise: RFC2183 & RFC2045 5.1
            // Fixes a warning in IETF's msglint MIME checker
            // Allow for bypassing the Content-Disposition header totally
            if (!(empty($disposition))) {
                if (preg_match('/[ \(\)<>@,;:\\"\/\[\]\?=]/', $name)) {
                    $mime[] = sprintf(
                        "Content-Disposition: %s; filename=\"%s\"%s",
                        $disposition,
                        $this->encodeHeader($this->secureHeader($name)),
                        $this->LE . $this->LE
                    );
                } else {
                    $mime[] = sprintf(
                        "Content-Disposition: %s; filename=%s%s",
                        $disposition,
                        $this->encodeHeader($this->secureHeader($name)),
                        $this->LE . $this->LE
                    );
                }
            } else {
                $mime[] = $this->LE;
            }

            // Encode as string attachment
            if ($bString) {
                $mime[] = $this->encodeString($string, $encoding);
                if ($this->isError()) {
                    return '';
                }
                $mime[] = $this->LE . $this->LE;
            } else {
```

```
                $mime[] = $this->encodeFile($path, $encoding);
                if ($this->isError()) {
                    return '';
                }
                $mime[] = $this->LE . $this->LE;
            }
        }
    }

    $mime[] = sprintf("--%s--%s", $boundary, $this->LE);

    return implode("", $mime);
}

/**
 * Encode a file attachment in requested format.
 * Returns an empty string on failure.
 * @param string $path The full path to the file
 * @param string $encoding The encoding to use; one of 'base64', '7bit', '8bit', 'binary', 'quoted-
printable'
 * @throws phpmailerException
 * @see EncodeFile(encodeFile
 * @access protected
 * @return string
 */
protected function encodeFile($path, $encoding = 'base64')
{
    try {
        if (!is_readable($path)) {
            throw new phpmailerException($this->lang('file_open') . $path, self::STOP_CONTINUE);
        }
        $magic_quotes = get_magic_quotes_runtime();
        if ($magic_quotes) {
            if (version_compare(PHP_VERSION, '5.3.0', '<')) {
                set_magic_quotes_runtime(0);
            } else {
                ini_set('magic_quotes_runtime', 0);
            }
        }
        $file_buffer = file_get_contents($path);
        $file_buffer = $this->encodeString($file_buffer, $encoding);
        if ($magic_quotes) {
            if (version_compare(PHP_VERSION, '5.3.0', '<')) {
                set_magic_quotes_runtime($magic_quotes);
            } else {
                ini_set('magic_quotes_runtime', $magic_quotes);
            }
        }
        return $file_buffer;
    } catch (Exception $e) {
        $this->setError($e->getMessage());
        return '';
    }
}

/**
 * Encode a string in requested format.
 * Returns an empty string on failure.
 * @param string $str The text to encode
 * @param string $encoding The encoding to use; one of 'base64', '7bit', '8bit', 'binary', 'quoted-
printable'
 * @access public
 * @return string
 */
public function encodeString($str, $encoding = 'base64')
{
    $encoded = '';
    switch (strtolower($encoding)) {
        case 'base64':
            $encoded = chunk_split(base64_encode($str), 76, $this->LE);
            break;
        case '7bit':
        case '8bit':
            $encoded = $this->fixEOL($str);
            //Make sure it ends with a line break
            if (substr($encoded, -(strlen($this->LE))) != $this->LE) {
                $encoded .= $this->LE;
            }
            break;
        case 'binary':
            $encoded = $str;
            break;
        case 'quoted-printable':
            $encoded = $this->encodeQP($str);
            break;
        default:
            $this->setError($this->lang('encoding') . $encoding);
            break;
    }
    return $encoded;
}

/**
 * Encode a header string optimally.
 * Picks shortest of Q, B, quoted-printable or none.
 * @access public
 * @param string $str
 * @param string $position
 * @return string
 */
public function encodeHeader($str, $position = 'text')
{
    $x = 0;
    switch (strtolower($position)) {
        case 'phrase':
            if (!preg_match('/[\200-\377]/', $str)) {
                // Can't use addslashes as we don't know what value has magic_quotes_sybase
                $encoded = addcslashes($str, "\0..\37\177\\\"");
                if (($str == $encoded) && !preg_match('/[^A-Za-z0-9!#$%&\'*+\/=?^_`{|}~ -]/', $str)) {
                    return ($encoded);
                } else {
                    return ("\"$encoded\"");
                }
            }
            $x = preg_match_all('/[^\040\041\043-\133\135-\176]/', $str, $matches);
            break;
        /** @noinspection PhpMissingBreakStatementInspection */
        case 'comment':
            $x = preg_match_all('/[()"]/', $str, $matches);
            // Intentional fall-through
        case 'text':
        default:
            $x += preg_match_all('/[\000-\010\013\014\016-\037\177-\377]/', $str, $matches);
            break;
    }
```

```
        if ($x == 0) { //There are no chars that need encoding
            return ($str);
        }

        $maxlen = 75 - 7 - strlen($this->CharSet);
        // Try to select the encoding which should produce the shortest output
        if ($x > strlen($str) / 3) {
            //More than a third of the content will need encoding, so B encoding will be most efficient
            $encoding = 'B';
            if (function_exists('mb_strlen') && $this->hasMultiBytes($str)) {
                // Use a custom function which correctly encodes and wraps long
                // multibyte strings without breaking lines within a character
                $encoded = $this->base64EncodeWrapMB($str, "\n");
            } else {
                $encoded = base64_encode($str);
                $maxlen -= $maxlen % 4;
                $encoded = trim(chunk_split($encoded, $maxlen, "\n"));
            }
        } else {
            $encoding = 'Q';
            $encoded = $this->encodeQ($str, $position);
            $encoded = $this->wrapText($encoded, $maxlen, true);
            $encoded = str_replace('=' . self::CRLF, "\n", trim($encoded));
        }

        $encoded = preg_replace('/^(.*)$/m', " =?" . $this->CharSet . "?$encoding?\\1?=", $encoded);
        $encoded = trim(str_replace("\n", $this->LE, $encoded));

        return $encoded;
    }

    /**
     * Check if a string contains multi-byte characters.
     * @access public
     * @param string $str multi-byte text to wrap encode
     * @return bool
     */
    public function hasMultiBytes($str)
    {
        if (function_exists('mb_strlen')) {
            return (strlen($str) > mb_strlen($str, $this->CharSet));
        } else { // Assume no multibytes (we can't handle without mbstring functions anyway)
```

```
            return false;
        }
    }

    /**
     * Encode and wrap long multibyte strings for mail headers
     * without breaking lines within a character.
     * Adapted from a function by paravoid at http://uk.php.net/manual/en/function.mb-encode-
mimeheader.php
     * @access public
     * @param string $str multi-byte text to wrap encode
     * @param string $lf string to use as linefeed/end-of-line
     * @return string
     */
    public function base64EncodeWrapMB($str, $lf = null)
    {
        $start = "=?" . $this->CharSet . "?B?";
        $end = "?=";
        $encoded = "";
        if ($lf === null) {
            $lf = $this->LE;
        }

        $mb_length = mb_strlen($str, $this->CharSet);
        // Each line must have length <= 75, including $start and $end
        $length = 75 - strlen($start) - strlen($end);
        // Average multi-byte ratio
        $ratio = $mb_length / strlen($str);
        // Base64 has a 4:3 ratio
        $avgLength = floor($length * $ratio * .75);

        for ($i = 0; $i < $mb_length; $i += $offset) {
            $lookBack = 0;
            do {
                $offset = $avgLength - $lookBack;
                $chunk = mb_substr($str, $i, $offset, $this->CharSet);
                $chunk = base64_encode($chunk);
                $lookBack++;
            } while (strlen($chunk) > $length);
            $encoded .= $chunk . $lf;
        }
        // Chomp the last linefeed
        $encoded = substr($encoded, 0, -strlen($lf));
        return $encoded;
    }

    /**
     * Encode a string in quoted-printable format.
     * According to RFC2045 section 6.7.
     * @access public
     * @param string $string The text to encode
     * @param integer $line_max Number of chars allowed on a line before wrapping
     * @return string
     * @link PHP version adapted from http://www.php.net/manual/en/function.quoted-printable-
decode.php#89417
     */
    public function encodeQP($string, $line_max = 76)
    {
        if (function_exists('quoted_printable_encode')) { //Use native function if it's available (>=
PHP5.3)
            return quoted_printable_encode($string);
        }
        //Fall back to a pure PHP implementation
        $string = str_replace(
            array('%20', '%0D%0A.', '%0D%0A', '%'),
            array(' ', "\r\n=2E", "\r\n", '='),
            rawurlencode($string)
        );
        $string = preg_replace('/[^\r\n]{' . ($line_max - 3) . '}[^=\r\n]{2}/', "$0=\r\n", $string);
        return $string;
    }

    /**
     * Backward compatibility wrapper for an old QP encoding function that was removed.
     * @see PHPMailer::encodeQP()
     * @access public
     * @param string $string
     * @param integer $line_max
     * @param bool $space_conv
     * @return string
     * @deprecated Use encodeQP instead.
     */
    public function encodeQPphp(
        $string,
        $line_max = 76,
        /** @noinspection PhpUnusedParameterInspection */ $space_conv = false
    ) {
        return $this->encodeQP($string, $line_max);
    }

    /**
     * Encode a string using Q encoding.
     * @link http://tools.ietf.org/html/rfc2047
     * @param string $str the text to encode
     * @param string $position Where the text is going to be used, see the RFC for what that
means
     * @access public
     * @return string
     */
    public function encodeQ($str, $position = 'text')
    {
        //There should not be any EOL in the string
        $pattern = '';
        $encoded = str_replace(array("\r", "\n"), '', $str);
        switch (strtolower($position)) {
            case 'phrase':
                //RFC 2047 section 5.3
                $pattern = '^A-Za-z0-9!*+\/ -';
                break;
            /** @noinspection PhpMissingBreakStatementInspection */
            case 'comment':
                //RFC 2047 section 5.2
                $pattern = '\(\)"';
                //intentional fall-through
                //for this reason we build the $pattern without including delimiters and []
            case 'text':
            default:
                //RFC 2047 section 5.1
                //Replace every high ascii, control, =, ? and _ characters
                $pattern = '\000-\011\013\014\016-\037\075\077\137\177-\377' . $pattern;
                break;
        }
        $matches = array();
        if (preg_match_all("/[{$pattern}]/", $encoded, $matches)) {
            //If the string contains an '=', make sure it's the first thing we replace
        //so as to avoid double-encoding
        $s = array_search('=', $matches[0]);
        if ($s !== false) {
            unset($matches[0][$s]);
            array_unshift($matches[0], '=');
        }
        foreach (array_unique($matches[0]) as $char) {
            $encoded = str_replace($char, '=' . sprintf('%02X', ord($char)), $encoded);
        }
    }
    //Replace every spaces to _ (more readable than =20)
    return str_replace(' ', '_', $encoded);
}


/**
 * Add a string or binary attachment (non-filesystem).
 * This method can be used to attach ascii or binary data,
 * such as a BLOB record from a database.
 * @param string $string String attachment data.
 * @param string $filename Name of the attachment.
 * @param string $encoding File encoding (see $Encoding).
 * @param string $type File extension (MIME) type.
 * @param string $disposition Disposition to use
 * @return void
 */
public function addStringAttachment(
    $string,
    $filename,
    $encoding = 'base64',
    $type = '',
    $disposition = 'attachment'
) {
    //If a MIME type is not specified, try to work it out from the file name
    if ($type == '') {
        $type = self::filenameToType($filename);
    }
    // Append to $attachment array
    $this->attachment[] = array(
        0 => $string,
        1 => $filename,
        2 => basename($filename),
            3 => $encoding,
            4 => $type,
            5 => true, // isStringAttachment
            6 => $disposition,
            7 => 0
        );
    }

    /**
     * Add an embedded (inline) attachment from a file.
     * This can include images, sounds, and just about any other document type.
     * These differ from 'regular' attachmants in that they are intended to be
     * displayed inline with the message, not just attached for download.
     * This is used in HTML messages that embed the images
     * the HTML refers to using the $cid value.
     * @param string $path Path to the attachment.
     * @param string $cid Content ID of the attachment; Use this to reference
     *       the content when using an embedded image in HTML.
     * @param string $name Overrides the attachment name.
     * @param string $encoding File encoding (see $Encoding).
     * @param string $type File MIME type.
     * @param string $disposition Disposition to use
     * @return bool True on successfully adding an attachment
     */
    public function addEmbeddedImage($path, $cid, $name = '', $encoding = 'base64', $type = '',
$disposition = 'inline')
    {
        if (!@is_file($path)) {
            $this->setError($this->lang('file_access') . $path);
            return false;
        }

        //If a MIME type is not specified, try to work it out from the file name
        if ($type == '') {
            $type = self::filenameToType($path);
        }

        $filename = basename($path);
        if ($name == '') {
            $name = $filename;
        }
    // Append to $attachment array
    $this->attachment[] = array(
        0 => $path,
        1 => $filename,
        2 => $name,
        3 => $encoding,
        4 => $type,
        5 => false, // isStringAttachment
        6 => $disposition,
        7 => $cid
    );
    return true;
}

/**
 * Add an embedded stringified attachment.
 * This can include images, sounds, and just about any other document type.
 * Be sure to set the $type to an image type for images:
 * JPEG images use 'image/jpeg', GIF uses 'image/gif', PNG uses 'image/png'.
 * @param string $string The attachment binary data.
 * @param string $cid Content ID of the attachment; Use this to reference
 *       the content when using an embedded image in HTML.
 * @param string $name
 * @param string $encoding File encoding (see $Encoding).
 * @param string $type MIME type.
 * @param string $disposition Disposition to use
 * @return bool True on successfully adding an attachment
 */
public function addStringEmbeddedImage(
    $string,
    $cid,
    $name = '',
    $encoding = 'base64',
    $type = '',
    $disposition = 'inline'
) {
    //If a MIME type is not specified, try to work it out from the name
    if ($type == '') {
        $type = self::filenameToType($name);
    }

    // Append to $attachment array
    $this->attachment[] = array(
        0 => $string,
        1 => $name,
        2 => $name,
        3 => $encoding,
        4 => $type,
        5 => true, // isStringAttachment
        6 => $disposition,
        7 => $cid
    );
    return true;
}

/**
 * Check if an inline attachment is present.
 * @access public
 * @return bool
 */
public function inlineImageExists()
{
    foreach ($this->attachment as $attachment) {
        if ($attachment[6] == 'inline') {
            return true;
        }
    }
    return false;
}

/**
 * Check if an attachment (non-inline) is present.
 * @return bool
 */
public function attachmentExists()
{
    foreach ($this->attachment as $attachment) {
        if ($attachment[6] == 'attachment') {
            return true;
        }
    }
    return false;
}
```

```
/**
 * Check if this message has an alternative body set.
 * @return bool
 */
public function alternativeExists()
{
    return !empty($this->AltBody);
}

/**
 * Clear all To recipients.
 * @return void
 */
public function clearAddresses()
{
    foreach ($this->to as $to) {
        unset($this->all_recipients[strtolower($to[0])]);
    }
    $this->to = array();
}

/**
 * Clear all CC recipients.
 * @return void
 */
public function clearCCs()
{
    foreach ($this->cc as $cc) {
        unset($this->all_recipients[strtolower($cc[0])]);
    }
    $this->cc = array();
}

/**
 * Clear all BCC recipients.
 * @return void
 */
public function clearBCCs()
{
    foreach ($this->bcc as $bcc) {
        unset($this->all_recipients[strtolower($bcc[0])]);
    }
    $this->bcc = array();
}

/**
 * Clear all ReplyTo recipients.
 * @return void
 */
public function clearReplyTos()
{
    $this->ReplyTo = array();
}

/**
 * Clear all recipient types.
 * @return void
 */
public function clearAllRecipients()
{
    $this->to = array();
    $this->cc = array();
    $this->bcc = array();
    $this->all_recipients = array();
}

/**
 * Clear all filesystem, string, and binary attachments.
 * @return void
 */
public function clearAttachments()
{
    $this->attachment = array();
}

/**
 * Clear all custom headers.
 * @return void
 */
public function clearCustomHeaders()
{
    $this->CustomHeader = array();
}
```

```
/**
 * Add an error message to the error container.
 * @access protected
 * @param string $msg
 * @return void
 */
protected function setError($msg)
{
    $this->error_count++;
    if ($this->Mailer == 'smtp' and !is_null($this->smtp)) {
        $lasterror = $this->smtp->getError();
        if (!empty($lasterror) and array_key_exists('smtp_msg', $lasterror)) {
            $msg .= '<p>' . $this->lang('smtp_error') . $lasterror['smtp_msg'] . "</p>\n";
        }
    }
    $this->ErrorInfo = $msg;
}

/**
 * Return an RFC 822 formatted date.
 * @access public
 * @return string
 * @static
 */
public static function rfcDate()
{
    //Set the time zone to whatever the default is to avoid 500 errors
    //Will default to UTC if it's not set properly in php.ini
    date_default_timezone_set(@date_default_timezone_get());
    return date('D, j M Y H:i:s O');
}

/**
 * Get the server hostname.
 * Returns 'localhost.localdomain' if unknown.
 * @access protected
 * @return string
 */
protected function serverHostname()
{
    if (!empty($this->Hostname)) {
        $result = $this->Hostname;
    } elseif (isset($_SERVER['SERVER_NAME'])) {
        $result = $_SERVER['SERVER_NAME'];
    } else {
        $result = 'localhost.localdomain';
    }

    return $result;
}

/**
 * Get an error message in the current language.
 * @access protected
 * @param string $key
 * @return string
 */
protected function lang($key)
{
    if (count($this->language) < 1) {
        $this->setLanguage('en'); // set the default language
    }

    if (isset($this->language[$key])) {
        return $this->language[$key];
    } else {
        return 'Language string failed to load: ' . $key;
    }
}

/**
 * Check if an error occurred.
 * @access public
 * @return bool True if an error did occur.
 */
public function isError()
{
    return ($this->error_count > 0);
}

/**
 * Ensure consistent line endings in a string.
 * Changes every end of line from CRLF, CR or LF to $this->LE.
 * @access public
 * @param string $str String to fixEOL
 * @return string
 */
public function fixEOL($str)
{
    // Normalise to \n
    $nstr = str_replace(array("\r\n", "\r"), "\n", $str);
    // Now convert LE as needed
    if ($this->LE !== "\n") {
        $nstr = str_replace("\n", $this->LE, $nstr);
    }
    return $nstr;
}

/**
 * Add a custom header.
 * $name value can be overloaded to contain
 * both header name and value (name:value)
 * @access public
 * @param string $name Custom header name
 * @param string $value Header value
 * @return void
 */
public function addCustomHeader($name, $value = null)
{
    if ($value === null) {
        // Value passed in as name:value
        $this->CustomHeader[] = explode(':', $name, 2);
    } else {
        $this->CustomHeader[] = array($name, $value);
    }
}

/**
 * Create a message from an HTML string.
 * Automatically makes modifications for inline images and backgrounds
 * and creates a plain-text version by converting the HTML.
 * Overwrites any existing values in $this->Body and $this->AltBody
 * @access public
 * @param string $message HTML message string
 * @param string $basedir baseline directory for path
 * @param bool $advanced Whether to use the advanced HTML to text converter
```

```
     * @return string $message
     */
    public function msgHTML($message, $basedir = '', $advanced = false)
    {
        preg_match_all("/(src|background)=[\"'](.*)[\"']/Ui", $message, $images);
        if (isset($images[2])) {
            foreach ($images[2] as $i => $url) {
                // do not change urls for absolute images (thanks to corvuscorax)
                if (!preg_match('#^[A-z]+://#', $url)) {
                    $filename = basename($url);
                    $directory = dirname($url);
                    if ($directory == '.') {
                        $directory = '';
                    }
                    $cid = md5($url) . '@phpmailer.0'; //RFC2392 S 2
                    if (strlen($basedir) > 1 && substr($basedir, -1) != '/') {
                        $basedir .= '/';
                    }
                    if (strlen($directory) > 1 && substr($directory, -1) != '/') {
                        $directory .= '/';
                    }
                    if ($this->addEmbeddedImage(
                        $basedir . $directory . $filename,
                        $cid,
                        $filename,
                        'base64',
                        self::_mime_types(self::mb_pathinfo($filename, PATHINFO_EXTENSION))
                    )
                    ) {
                        $message = preg_replace(
                            "/" . $images[1][$i] . "=[\"']" . preg_quote($url, '/') . "[\"']/Ui",
                            $images[1][$i] . "=\"cid:" . $cid . "\"",
                            $message
                        );
                    }
                }
            }
        }
        $this->isHTML(true);
        if (empty($this->AltBody)) {
            $this->AltBody = 'To view this email message, open it in a program that understands
HTML!' . "\n\n";
```

```
        }
        //Convert all message body line breaks to CRLF, makes quoted-printable encoding work
much better
        $this->Body = $this->normalizeBreaks($message);
        $this->AltBody = $this->normalizeBreaks($this->html2text($message, $advanced));
        return $this->Body;
    }

    /**
     * Convert an HTML string into plain text.
     * @param string $html The HTML text to convert
     * @param bool $advanced Should this use the more complex html2text converter or just a
simple one?
     * @return string
     */
    public function html2text($html, $advanced = false)
    {
        if ($advanced) {
            require_once 'extras/class.html2text.php';
            $h = new html2text($html);
            return $h->get_text();
        }
        return html_entity_decode(
            trim(strip_tags(preg_replace('/<(head|title|style|script)[^>]*>.*?<\/\\1>/si', '', $html))),
            ENT_QUOTES,
            $this->CharSet
        );
    }

    /**
     * Get the MIME type for a file extension.
     * @param string $ext File extension
     * @access public
     * @return string MIME type of file.
     * @static
     */
    public static function _mime_types($ext = '')
    {
        $mimes = array(
            'xl' => 'application/excel',
            'hqx' => 'application/mac-binhex40',
            'cpt' => 'application/mac-compactpro',
            'bin' => 'application/macbinary',
            'doc' => 'application/msword',
            'word' => 'application/msword',
            'class' => 'application/octet-stream',
            'dll' => 'application/octet-stream',
            'dms' => 'application/octet-stream',
            'exe' => 'application/octet-stream',
            'lha' => 'application/octet-stream',
            'lzh' => 'application/octet-stream',
            'psd' => 'application/octet-stream',
            'sea' => 'application/octet-stream',
            'so' => 'application/octet-stream',
            'oda' => 'application/oda',
            'pdf' => 'application/pdf',
            'ai' => 'application/postscript',
            'eps' => 'application/postscript',
            'ps' => 'application/postscript',
            'smi' => 'application/smil',
            'smil' => 'application/smil',
            'mif' => 'application/vnd.mif',
            'xls' => 'application/vnd.ms-excel',
            'ppt' => 'application/vnd.ms-powerpoint',
            'wbxml' => 'application/vnd.wap.wbxml',
            'wmlc' => 'application/vnd.wap.wmlc',
            'dcr' => 'application/x-director',
            'dir' => 'application/x-director',
            'dxr' => 'application/x-director',
            'dvi' => 'application/x-dvi',
            'gtar' => 'application/x-gtar',
            'php3' => 'application/x-httpd-php',
            'php4' => 'application/x-httpd-php',
            'php' => 'application/x-httpd-php',
            'phtml' => 'application/x-httpd-php',
            'phps' => 'application/x-httpd-php-source',
            'js' => 'application/x-javascript',
            'swf' => 'application/x-shockwave-flash',
            'sit' => 'application/x-stuffit',
            'tar' => 'application/x-tar',
            'tgz' => 'application/x-tar',
            'xht' => 'application/xhtml+xml',
            'xhtml' => 'application/xhtml+xml',
            'zip' => 'application/zip',
            'mid' => 'audio/midi',
            'midi' => 'audio/midi',
            'mp2' => 'audio/mpeg',
            'mp3' => 'audio/mpeg',
            'mpga' => 'audio/mpeg',
            'aif' => 'audio/x-aiff',
            'aifc' => 'audio/x-aiff',
            'aiff' => 'audio/x-aiff',
            'ram' => 'audio/x-pn-realaudio',
            'rm' => 'audio/x-pn-realaudio',
            'rpm' => 'audio/x-pn-realaudio-plugin',
            'ra' => 'audio/x-realaudio',
            'wav' => 'audio/x-wav',
            'bmp' => 'image/bmp',
            'gif' => 'image/gif',
            'jpeg' => 'image/jpeg',
            'jpe' => 'image/jpeg',
            'jpg' => 'image/jpeg',
            'png' => 'image/png',
            'tiff' => 'image/tiff',
            'tif' => 'image/tiff',
            'eml' => 'message/rfc822',
            'css' => 'text/css',
            'html' => 'text/html',
            'htm' => 'text/html',
            'shtml' => 'text/html',
            'log' => 'text/plain',
            'text' => 'text/plain',
            'txt' => 'text/plain',
            'rtx' => 'text/richtext',
            'rtf' => 'text/rtf',
            'xml' => 'text/xml',
            'xsl' => 'text/xml',
            'mpeg' => 'video/mpeg',
            'mpe' => 'video/mpeg',
            'mpg' => 'video/mpeg',
            'mov' => 'video/quicktime',
            'qt' => 'video/quicktime',
            'rv' => 'video/vnd.rn-realvideo',
            'avi' => 'video/x-msvideo',
            'movie' => 'video/x-sgi-movie'
        );
```

```
        return (array_key_exists(strtolower($ext), $mimes) ? $mimes[strtolower($ext)]:
'application/octet-stream');
    }

    /**
     * Map a file name to a MIME type.
     * Defaults to 'application/octet-stream', i.e.. arbitrary binary data.
     * @param string $filename A file name or full path, does not need to exist as a file
     * @return string
     * @static
     */
    public static function filenameToType($filename)
    {
        //In case the path is a URL, strip any query string before getting extension
        $qpos = strpos($filename, '?');
        if ($qpos !== false) {
            $filename = substr($filename, 0, $qpos);
        }
        $pathinfo = self::mb_pathinfo($filename);
        return self::_mime_types($pathinfo['extension']);
    }

    /**
     * Multi-byte-safe pathinfo replacement.
     * Drop-in replacement for pathinfo(), but multibyte-safe, cross-platform-safe, old-version-safe.
     * Works similarly to the one in PHP >= 5.2.0
     * @link http://www.php.net/manual/en/function.pathinfo.php#107461
     * @param string $path A filename or path, does not need to exist as a file
     * @param integer|string $options Either a PATHINFO_* constant,
     *      or a string name to return only the specified piece, allows 'filename' to work on PHP < 5.2
     * @return string|array
     * @static
     */
    public static function mb_pathinfo($path, $options = null)
    {
        $ret = array('dirname' => '', 'basename' => '', 'extension' => '', 'filename' => '');
        $m = array();
        preg_match('%^(.*?)[\\\\/]*(([^/\\\\]*?)(\.([^\.\\\\/]+?)|))[\\\\/\.]*$%im', $path, $m);
        if (array_key_exists(1, $m)) {
            $ret['dirname'] = $m[1];
        }
        if (array_key_exists(2, $m)) {
```

```
        $ret['basename'] = $m[2];
    }
    if (array_key_exists(5, $m)) {
        $ret['extension'] = $m[5];
    }
    if (array_key_exists(3, $m)) {
        $ret['filename'] = $m[3];
    }
    switch ($options) {
        case PATHINFO_DIRNAME:
        case 'dirname':
            return $ret['dirname'];
            break;
        case PATHINFO_BASENAME:
        case 'basename':
            return $ret['basename'];
            break;
        case PATHINFO_EXTENSION:
        case 'extension':
            return $ret['extension'];
            break;
        case PATHINFO_FILENAME:
        case 'filename':
            return $ret['filename'];
            break;
        default:
            return $ret;
    }
}

/**
 * Set or reset instance properties.
 *
 * Usage Example:
 * $page->set('X-Priority', '3');
 *
 * @access public
 * @param string $name
 * @param mixed $value
 * NOTE: will not work with arrays, there are no arrays to set/reset
 * @throws phpmailerException
 * @return bool
```

```
     * @todo Should this not be using __set() magic function?
     */
    public function set($name, $value = '')
    {
        try {
            if (isset($this->$name)) {
                $this->$name = $value;
            } else {
                throw new phpmailerException($this->lang('variable_set') . $name,
self::STOP_CRITICAL);
            }
        } catch (Exception $e) {
            $this->setError($e->getMessage());
            if ($e->getCode() == self::STOP_CRITICAL) {
                return false;
            }
        }
        return true;
    }

    /**
     * Strip newlines to prevent header injection.
     * @access public
     * @param string $str
     * @return string
     */
    public function secureHeader($str)
    {
        return trim(str_replace(array("\r", "\n"), '', $str));
    }

    /**
     * Normalize line breaks in a string.
     * Converts UNIX LF, Mac CR and Windows CRLF line breaks into a single line break format.
     * Defaults to CRLF (for message bodies) and preserves consecutive breaks.
     * @param string $text
     * @param string $breaktype What kind of line break to use, defaults to CRLF
     * @return string
     * @access public
     * @static
     */
    public static function normalizeBreaks($text, $breaktype = "\r\n")
    {
        return preg_replace('/(\r\n|\r|\n)/ms', $breaktype, $text);
    }


    /**
     * Set the public and private key files and password for S/MIME signing.
     * @access public
     * @param string $cert_filename
     * @param string $key_filename
     * @param string $key_pass Password for private key
     */
    public function sign($cert_filename, $key_filename, $key_pass)
    {
        $this->sign_cert_file = $cert_filename;
        $this->sign_key_file = $key_filename;
        $this->sign_key_pass = $key_pass;
    }

    /**
     * Quoted-Printable-encode a DKIM header.
     * @access public
     * @param string $txt
     * @return string
     */
    public function DKIM_QP($txt)
    {
        $line = '';
        for ($i = 0; $i < strlen($txt); $i++) {
            $ord = ord($txt[$i]);
            if (((0x21 <= $ord) && ($ord <= 0x3A)) || $ord == 0x3C || ((0x3E <= $ord) && ($ord <=
0x7E))) {
                $line .= $txt[$i];
            } else {
                $line .= "=" . sprintf("%02X", $ord);
            }
        }
        return $line;
    }

    /**
     * Generate a DKIM signature.
     * @access public
     * @param string $s Header
     * @throws phpmailerException
     * @return string
     */
    public function DKIM_Sign($s)
    {
        if (!defined('PKCS7_TEXT')) {
            if ($this->exceptions) {
                throw new phpmailerException($this->lang("signing") . ' OpenSSL extension missing.');
            }
            return '';
        }
        $privKeyStr = file_get_contents($this->DKIM_private);
        if ($this->DKIM_passphrase != '') {
            $privKey = openssl_pkey_get_private($privKeyStr, $this->DKIM_passphrase);
        } else {
            $privKey = $privKeyStr;
        }
        if (openssl_sign($s, $signature, $privKey)) {
            return base64_encode($signature);
        }
        return '';
    }

    /**
     * Generate a DKIM canonicalization header.
     * @access public
     * @param string $s Header
     * @return string
     */
    public function DKIM_HeaderC($s)
    {
        $s = preg_replace("/\r\n\s+/", " ", $s);
        $lines = explode("\r\n", $s);
        foreach ($lines as $key => $line) {
            list($heading, $value) = explode(":", $line, 2);
            $heading = strtolower($heading);
            $value = preg_replace("/\s+/", " ", $value); // Compress useless spaces
            $lines[$key] = $heading . ":" . trim($value); // Don't forget to remove WSP around the value
        }
        $s = implode("\r\n", $lines);
        return $s;
    }

    /**
     * Generate a DKIM canonicalization body.
     * @access public
     * @param string $body Message Body
     * @return string
     */
    public function DKIM_BodyC($body)
    {
        if ($body == '') {
            return "\r\n";
        }
        // stabilize line endings
        $body = str_replace("\r\n", "\n", $body);
        $body = str_replace("\n", "\r\n", $body);
        // END stabilize line endings
        while (substr($body, strlen($body) - 4, 4) == "\r\n\r\n") {
            $body = substr($body, 0, strlen($body) - 2);
        }
        return $body;
    }

    /**
     * Create the DKIM header and body in a new message header.
     * @access public
     * @param string $headers_line Header lines
     * @param string $subject Subject
     * @param string $body Body
     * @return string
     */
    public function DKIM_Add($headers_line, $subject, $body)
    {
        $DKIMsignatureType = 'rsa-sha1'; // Signature & hash algorithms
        $DKIMcanonicalization = 'relaxed/simple'; // Canonicalization of header/body
        $DKIMquery = 'dns/txt'; // Query method
        $DKIMtime = time(); // Signature Timestamp = seconds since 00:00:00 - Jan 1, 1970 (UTC
time zone)
        $subject_header = "Subject: $subject";
        $headers = explode($this->LE, $headers_line);
        $from_header = '';
        $to_header = '';
        $current = '';
        foreach ($headers as $header) {
            if (strpos($header, 'From:') === 0) {
                $from_header = $header;
                $current = 'from_header';
            } elseif (strpos($header, 'To:') === 0) {
                $to_header = $header;
                $current = 'to_header';
            } else {
                if ($current && strpos($header, ' =?') === 0) {
                    $current .= $header;
                } else {
                    $current = '';
                }
            }
        }
        $from = str_replace('|', '=7C', $this->DKIM_QP($from_header));
        $to = str_replace('|', '=7C', $this->DKIM_QP($to_header));
        $subject = str_replace(
            '|',
            '=7C',
            $this->DKIM_QP($subject_header)
        ); // Copied header fields (dkim-quoted-printable)
        $body = $this->DKIM_BodyC($body);
        $DKIMlen = strlen($body); // Length of body
        $DKIMb64 = base64_encode(pack("H*", sha1($body))); // Base64 of packed binary SHA-1
hash of body
        $ident = ($this->DKIM_identity == '') ? '' : ' i=' . $this->DKIM_identity . ';';
        $dkimhdrs = "DKIM-Signature: v=1; a=" .
            $DKIMsignatureType . "; q=" .
            $DKIMquery . "; l=" .
            $DKIMlen . "; s=" .
            $this->DKIM_selector .
            ";\r\n" .
            "\tt=" . $DKIMtime . "; c=" . $DKIMcanonicalization . ";\r\n" .
            "\th=From:To:Subject;\r\n" .
            "\td=" . $this->DKIM_domain . ';' . $ident . "\r\n" .
            "\tz=$from\r\n" .
            "\t|$to\r\n" .
            "\t|$subject;\r\n" .
            "\tbh=" . $DKIMb64 . ";\r\n" .
            "\tb=";
        $toSign = $this->DKIM_HeaderC(
            $from_header . "\r\n" . $to_header . "\r\n" . $subject_header . "\r\n" . $dkimhdrs
        );
        $signed = $this->DKIM_Sign($toSign);
        return $dkimhdrs . $signed . "\r\n";
    }

    /**
     * Perform a callback.
     * @param bool $isSent
     * @param string $to
     * @param string $cc
     * @param string $bcc
     * @param string $subject
     * @param string $body
     * @param string $from
     */
    protected function doCallback($isSent, $to, $cc, $bcc, $subject, $body, $from = null)
    {
        if (!empty($this->action_function) && is_callable($this->action_function)) {
            $params = array($isSent, $to, $cc, $bcc, $subject, $body, $from);
            call_user_func_array($this->action_function, $params);
        }
    }
}

/**
 * PHPMailer exception handler
 * @package PHPMailer
 */
class phpmailerException extends Exception
{
    /**
     * Prettify error message output
     * @return string
     */
    public function errorMessage()
    {
        $errorMsg = '<strong>' . $this->getMessage() . "</strong><br />\n";
        return $errorMsg;
    }
```

```
}

66:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\class.pop3.php
<?php
/**
 * PHPMailer POP-Before-SMTP Authentication Class.
 * PHP Version 5.0.0
 * Version 5.2.7
 * @package PHPMailer
 * @link https://github.com/PHPMailer/PHPMailer/
 * @author Marcus Bointon (coolbru) <phpmailer@synchromedia.co.uk>
 * @author Jim Jagielski (jimjag) <jimjag@gmail.com>
 * @author Andy Prevost (codeworxtech) <codeworxtech@users.sourceforge.net>
 * @author Brent R. Matzelle (original founder)
 * @copyright 2013 Marcus Bointon
 * @copyright 2010 - 2012 Jim Jagielski
 * @copyright 2004 - 2009 Andy Prevost
 * @license http://www.gnu.org/copyleft/lesser.html GNU Lesser General Public License
 * @note This program is distributed in the hope that it will be useful - WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.
 */

/**
 * PHPMailer POP-Before-SMTP Authentication Class.
 * Specifically for PHPMailer to use for RFC1939 POP-before-SMTP authentication.
 * Does not support APOP.
 * @package PHPMailer
 * @author Richard Davey (original author) <rich@corephp.co.uk>
 * @author Marcus Bointon (coolbru) <phpmailer@synchromedia.co.uk>
 * @author Jim Jagielski (jimjag) <jimjag@gmail.com>
 * @author Andy Prevost (codeworxtech) <codeworxtech@users.sourceforge.net>
 */

class POP3
{
    /**
     * The POP3 PHPMailer Version number.
     * @type string
     * @access public
     */
    public $Version = '5.2.7';
```

```
/**
 * Default POP3 port number.
 * @type int
 * @access public
 */
public $POP3_PORT = 110;

/**
 * Default timeout in seconds.
 * @type int
 * @access public
 */
public $POP3_TIMEOUT = 30;

/**
 * POP3 Carriage Return + Line Feed.
 * @type string
 * @access public
 * @deprecated Use the constant instead
 */
public $CRLF = "\r\n";

/**
 * Debug display level.
 * Options: 0 = no, 1+ = yes
 * @type int
 * @access public
 */
public $do_debug = 0;

/**
 * POP3 mail server hostname.
 * @type string
 * @access public
 */
public $host;

/**
 * POP3 port number.
 * @type int
 * @access public
 */
public $port;

/**
 * POP3 Timeout Value in seconds.
 * @type int
 * @access public
 */
public $tval;

/**
 * POP3 username
 * @type string
 * @access public
 */
public $username;

/**
 * POP3 password.
 * @type string
 * @access public
 */
public $password;

/**
 * Resource handle for the POP3 connection socket.
 * @type resource
 * @access private
 */
private $pop_conn;

/**
 * Are we connected?
 * @type bool
 * @access private
 */
private $connected;

/**
 * Error container.
 * @type array
 * @access private
 */
private $error;

/**
 * Line break constant
 */
const CRLF = "\r\n";

/**
 * Constructor.
 * @access public
 */
public function __construct()
{
    $this->pop_conn = 0;
    $this->connected = false;
    $this->error = null;
}

/**
 * Simple static wrapper for all-in-one POP before SMTP
 * @param $host
 * @param bool $port
 * @param bool $tval
 * @param string $username
 * @param string $password
 * @param int $debug_level
 * @return bool
 */
public static function popBeforeSmtp(
    $host,
    $port = false,
    $tval = false,
    $username = '',
    $password = '',
    $debug_level = 0
) {
    $pop = new POP3;
    return $pop->authorise($host, $port, $tval, $username, $password, $debug_level);
}

/**
```

```
     * Authenticate with a POP3 server.
     * A connect, login, disconnect sequence
     * appropriate for POP-before SMTP authorisation.
     * @access public
     * @param string $host
     * @param bool|int $port
     * @param bool|int $tval
     * @param string $username
     * @param string $password
     * @param int $debug_level
     * @return bool
     */
    public function authorise($host, $port = false, $tval = false, $username = '', $password = '',
$debug_level = 0)
    {
        $this->host = $host;
        // If no port value provided, use default
        if ($port === false) {
            $this->port = $this->POP3_PORT;
        } else {
            $this->port = $port;
        }
        // If no timeout value provided, use default
        if ($tval === false) {
            $this->tval = $this->POP3_TIMEOUT;
        } else {
            $this->tval = $tval;
        }
        $this->do_debug = $debug_level;
        $this->username = $username;
        $this->password = $password;
        //  Refresh the error log
        $this->error = null;
        //  connect
        $result = $this->connect($this->host, $this->port, $this->tval);
        if ($result) {
            $login_result = $this->login($this->username, $this->password);
            if ($login_result) {
                $this->disconnect();
                return true;
            }
        }
        // We need to disconnect regardless of whether the login succeeded
        $this->disconnect();
        return false;
    }

    /**
     * Connect to a POP3 server.
     * @access public
     * @param string $host
     * @param bool|int $port
     * @param integer $tval
     * @return boolean
     */
    public function connect($host, $port = false, $tval = 30)
    {
        //  Are we already connected?
        if ($this->connected) {
            return true;
        }

        //On Windows this will raise a PHP Warning error if the hostname doesn't exist.
        //Rather than suppress it with @fsockopen, capture it cleanly instead
        set_error_handler(array($this, 'catchWarning'));

        //  connect to the POP3 server
        $this->pop_conn = fsockopen(
            $host, //  POP3 Host
            $port, //  Port #
            $errno, //  Error Number
            $errstr, //  Error Message
            $tval
        ); //  Timeout (seconds)
        //  Restore the error handler
        restore_error_handler();
        //  Does the Error Log now contain anything?
        if ($this->error && $this->do_debug >= 1) {
            $this->displayErrors();
        }
        //  Did we connect?
        if ($this->pop_conn == false) {
            //  It would appear not...
            $this->error = array(
                'error' => "Failed to connect to server $host on port $port",
                'errno' => $errno,
                'errstr' => $errstr
            );
            if ($this->do_debug >= 1) {
                $this->displayErrors();
            }
            return false;
        }

        //  Increase the stream time-out
        //  Check for PHP 4.3.0 or later
        if (version_compare(phpversion(), '5.0.0', 'ge')) {
            stream_set_timeout($this->pop_conn, $tval, 0);
        } else {
            //  Does not work on Windows
            if (substr(PHP_OS, 0, 3) !== 'WIN') {
                socket_set_timeout($this->pop_conn, $tval, 0);
            }
        }

        //  Get the POP3 server response
        $pop3_response = $this->getResponse();
        //  Check for the +OK
        if ($this->checkResponse($pop3_response)) {
            //  The connection is established and the POP3 server is talking
            $this->connected = true;
            return true;
        }
        return false;
    }

    /**
     * Log in to the POP3 server.
     * Does not support APOP (RFC 2828, 4949).
     * @access public
     * @param string $username
     * @param string $password
     * @return boolean
     */
    public function login($username = '', $password = '')
    {
```

```
    if ($this->connected == false) {
        $this->error = 'Not connected to POP3 server';

        if ($this->do_debug >= 1) {
            $this->displayErrors();
        }
    }
    if (empty($username)) {
        $username = $this->username;
    }
    if (empty($password)) {
        $password = $this->password;
    }

    // Send the Username
    $this->sendString("USER $username" . self::CRLF);
    $pop3_response = $this->getResponse();
    if ($this->checkResponse($pop3_response)) {
        // Send the Password
        $this->sendString("PASS $password" . self::CRLF);
        $pop3_response = $this->getResponse();
        if ($this->checkResponse($pop3_response)) {
            return true;
        }
    }
    return false;
}

/**
 * Disconnect from the POP3 server.
 * @access public
 */
public function disconnect()
{
    $this->sendString('QUIT');
    //The QUIT command may cause the daemon to exit, which will kill our connection
    //So ignore errors here
    @fclose($this->pop_conn);
}

/**
 * Get a response from the POP3 server.
 * $size is the maximum number of bytes to retrieve
 * @param integer $size
 * @return string
 * @access private
 */
private function getResponse($size = 128)
{
    $r = fgets($this->pop_conn, $size);
    if ($this->do_debug >= 1) {
        echo "Server -> Client: $r";
    }
    return $r;
}

/**
 * Send raw data to the POP3 server.
 * @param string $string
 * @return integer
 * @access private
 */
private function sendString($string)
{
    if ($this->pop_conn) {
        if ($this->do_debug >= 2) { //Show client messages when debug >= 2
            echo "Client -> Server: $string";
        }
        return fwrite($this->pop_conn, $string, strlen($string));
    }
    return 0;
}

/**
 * Checks the POP3 server response.
 * Looks for for +OK or -ERR.
 * @param string $string
 * @return boolean
 * @access private
 */
private function checkResponse($string)
{
    if (substr($string, 0, 3) !== '+OK') {
        $this->error = array(
            'error' => "Server reported an error: $string",
            'errno' => 0,
            'errstr' => ''
        );
        if ($this->do_debug >= 1) {
            $this->displayErrors();
        }
        return false;
    } else {
        return true;
    }
}

/**
 * Display errors if debug is enabled.
 * @access private
 */
private function displayErrors()
{
    echo '<pre>';
    foreach ($this->error as $single_error) {
        print_r($single_error);
    }
    echo '</pre>';
}

/**
 * POP3 connection error handler.
 * @param integer $errno
 * @param string $errstr
 * @param string $errfile
 * @param integer $errline
 * @access private
 */
private function catchWarning($errno, $errstr, $errfile, $errline)
{
    $this->error[] = array(
        'error' => "Connecting to the POP3 server raised a PHP warning: ",
        'errno' => $errno,
        'errstr' => $errstr,
        'errfile' => $errfile,
        'errline' => $errline
        );
    }
}

67:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\class.smtp.php
<?php
/**
 * PHPMailer RFC821 SMTP email transport class.
 * Version 5.2.7
 * PHP version 5.0.0
 * @category  PHP
 * @package   PHPMailer
 * @link      https://github.com/PHPMailer/PHPMailer/
 * @author Marcus Bointon (coolbru) <phpmailer@synchromedia.co.uk>
 * @author Jim Jagielski (jimjag) <jimjag@gmail.com>
 * @author Andy Prevost (codeworxtech) <codeworxtech@users.sourceforge.net>
 * @copyright 2013 Marcus Bointon
 * @copyright 2004 - 2008 Andy Prevost
 * @copyright 2010 - 2012 Jim Jagielski
 * @license   http://www.gnu.org/copyleft/lesser.html Distributed under the Lesser General Public
License (LGPL)
 */

/**
 * PHPMailer RFC821 SMTP email transport class.
 *
 * Implements RFC 821 SMTP commands
 * and provides some utility methods for sending mail to an SMTP server.
 *
 * PHP Version 5.0.0
 *
 * @category PHP
 * @package  PHPMailer
 * @link     https://github.com/PHPMailer/PHPMailer/blob/master/class.smtp.php
 * @author   Chris Ryan <unknown@example.com>
 * @author   Marcus Bointon <phpmailer@synchromedia.co.uk>
 * @license  http://www.gnu.org/copyleft/lesser.html Distributed under the Lesser General Public
License (LGPL)
 */

class SMTP
{
```

```
/**
 * The PHPMailer SMTP Version number.
 */
const VERSION = '5.2.7';

/**
 * SMTP line break constant.
 */
const CRLF = "\r\n";

/**
 * The SMTP port to use if one is not specified.
 */
const DEFAULT_SMTP_PORT = 25;

/**
 * The PHPMailer SMTP Version number.
 * @type string
 * @deprecated This should be a constant
 * @see SMTP::VERSION
 */
public $Version = '5.2.7';

/**
 * SMTP server port number.
 * @type int
 * @deprecated This is only ever ued as default value, so should be a constant
 * @see SMTP::DEFAULT_SMTP_PORT
 */
public $SMTP_PORT = 25;

/**
 * SMTP reply line ending
 * @type string
 * @deprecated Use the class constant instead
 * @see SMTP::CRLF
 */
public $CRLF = "\r\n";

/**
 * Debug output level.
 * Options:
```

```
 *   0: no output
 *   1: commands
 *   2: data and commands
 *   3: as 2 plus connection status
 *   4: low level data output
 * @type int
 */
public $do_debug = 0;

/**
 * The function/method to use for debugging output.
 * Options: 'echo', 'html' or 'error_log'
 * @type string
 */
public $Debugoutput = 'echo';

/**
 * Whether to use VERP.
 * @type bool
 */
public $do_verp = false;

/**
 * The timeout value for connection, in seconds.
 * Default of 5 minutes (300sec) is from RFC2821 section 4.5.3.2
 * @type int
 */
public $Timeout = 300;

/**
 * The SMTP timelimit value for reads, in seconds.
 * @type int
 */
public $Timelimit = 30;

/**
 * The socket for the server connection.
 * @type resource
 */
protected $smtp_conn;

/**
```

```
 * Error message, if any, for the last call.
 * @type string
 */
protected $error = '';

/**
 * The reply the server sent to us for HELO.
 * @type string
 */
protected $helo_rply = '';

/**
 * The most recent reply received from the server.
 * @type string
 */
protected $last_reply = '';

/**
 * Constructor.
 * @access public
 */
public function __construct()
{
    $this->smtp_conn = 0;
    $this->error = null;
    $this->helo_rply = null;

    $this->do_debug = 0;
}

/**
 * Output debugging info via a user-selected method.
 * @param string $str Debug string to output
 * @return void
 */
protected function edebug($str)
{
    switch ($this->Debugoutput) {
        case 'error_log':
            //Don't output, just log
            error_log($str);
            break;
        case 'html':
            //Cleans up output a bit for a better looking, HTML-safe output
            echo htmlentities(
                preg_replace('/[\r\n]+/', '', $str),
                ENT_QUOTES,
                'UTF-8'
            )
            . "<br>\n";
            break;
        case 'echo':
        default:
            echo gmdate('Y-m-d H:i:s')."\t".trim($str)."\n";
    }
}

/**
 * Connect to an SMTP server.
 * @param string $host    SMTP server IP or host name
 * @param int $port    The port number to connect to
 * @param int $timeout How long to wait for the connection to open
 * @param array $options An array of options for stream_context_create()
 * @access public
 * @return bool
 */
public function connect($host, $port = null, $timeout = 30, $options = array())
{
    // Clear errors to avoid confusion
    $this->error = null;

    // Make sure we are __not__ connected
    if ($this->connected()) {
        // Already connected, generate error
        $this->error = array('error' => 'Already connected to a server');
        return false;
    }

    if (empty($port)) {
        $port = self::DEFAULT_SMTP_PORT;
    }

    // Connect to the SMTP server
    if ($this->do_debug >= 3) {
```

```
    $this->edebug('Connection: opening');
}

$errno = 0;
$errstr = '';
$socket_context = stream_context_create($options);
//Suppress errors; connection failures are handled at a higher level
$this->smtp_conn = @stream_socket_client(
    $host . ":" . $port,
    $errno,
    $errstr,
    $timeout,
    STREAM_CLIENT_CONNECT,
    $socket_context
);

// Verify we connected properly
if (empty($this->smtp_conn)) {
    $this->error = array(
        'error' => 'Failed to connect to server',
        'errno' => $errno,
        'errstr' => $errstr
    );
    if ($this->do_debug >= 1) {
        $this->edebug(
            'SMTP ERROR: ' . $this->error['error']
            . ": $errstr ($errno)"
        );
    }
    return false;
}
if ($this->do_debug >= 3) {
    $this->edebug('Connection: opened');
}

// SMTP server can take longer to respond, give longer timeout for first read
// Windows does not have support for this timeout function
if (substr(PHP_OS, 0, 3) != 'WIN') {
    $max = ini_get('max_execution_time');
    if ($max != 0 && $timeout > $max) { // Don't bother if unlimited
        @set_time_limit($timeout);
    }
```

```
        stream_set_timeout($this->smtp_conn, $timeout, 0);
    }

    // Get any announcement
    $announce = $this->get_lines();

    if ($this->do_debug >= 2) {
        $this->edebug('SERVER -> CLIENT: ' . $announce);
    }

    return true;
}

/**
 * Initiate a TLS (encrypted) session.
 * @access public
 * @return bool
 */
public function startTLS()
{
    if (!$this->sendCommand("STARTTLS", "STARTTLS", 220)) {
        return false;
    }
    // Begin encrypted connection
    if (!stream_socket_enable_crypto(
        $this->smtp_conn,
        true,
        STREAM_CRYPTO_METHOD_TLS_CLIENT
    )
    ) {
        return false;
    }
    return true;
}

/**
 * Perform SMTP authentication.
 * Must be run after hello().
 * @see hello()
 * @param string $username    The user name
 * @param string $password    The password
 * @param string $authtype    The auth type (PLAIN, LOGIN, NTLM, CRAM-MD5)
 * @param string $realm       The auth realm for NTLM
 * @param string $workstation The auth workstation for NTLM
 * @access public
 * @return bool True if successfully authenticated.
 */
public function authenticate(
    $username,
    $password,
    $authtype = 'LOGIN',
    $realm = '',
    $workstation = ''
) {
    if (empty($authtype)) {
        $authtype = 'LOGIN';
    }

    switch ($authtype) {
        case 'PLAIN':
            // Start authentication
            if (!$this->sendCommand('AUTH', 'AUTH PLAIN', 334)) {
                return false;
            }
            // Send encoded username and password
            if (!$this->sendCommand(
                'User & Password',
                base64_encode("\0" . $username . "\0" . $password),
                235
            )
            ) {
                return false;
            }
            break;
        case 'LOGIN':
            // Start authentication
            if (!$this->sendCommand('AUTH', 'AUTH LOGIN', 334)) {
                return false;
            }
            if (!$this->sendCommand("Username", base64_encode($username), 334)) {
                return false;
            }
            if (!$this->sendCommand("Password", base64_encode($password), 235)) {
                return false;
            }
            break;
        case 'NTLM':
            /*
             * ntlm_sasl_client.php
             * Bundled with Permission
             *
             * How to telnet in windows:
             * http://technet.microsoft.com/en-us/library/aa995718%28EXCHG.65%29.aspx
             * PROTOCOL Docs http://curl.haxx.se/rfc/ntlm.html#ntlmSmtpAuthentication
             */
            require_once 'extras/ntlm_sasl_client.php';
            $temp = new stdClass();
            $ntlm_client = new ntlm_sasl_client_class;
            //Check that functions are available
            if (!$ntlm_client->Initialize($temp)) {
                $this->error = array('error' => $temp->error);
                if ($this->do_debug >= 1) {
                    $this->edebug(
                        'You need to enable some modules in your php.ini file: '
                        . $this->error['error']
                    );
                }
                return false;
            }
            //msg1
            $msg1 = $ntlm_client->TypeMsg1($realm, $workstation); //msg1

            if (!$this->sendCommand(
                'AUTH NTLM',
                'AUTH NTLM ' . base64_encode($msg1),
                334
            )
            ) {
                return false;
            }

            //Though 0 based, there is a white space after the 3 digit number
            //msg2
            $challenge = substr($this->last_reply, 3);
            $challenge = base64_decode($challenge);
            $ntlm_res = $ntlm_client->NTLMResponse(
                substr($challenge, 24, 8),
                $password
            );
            //msg3
            $msg3 = $ntlm_client->TypeMsg3(
                $ntlm_res,
                $username,
                $realm,
                $workstation
            );
            // send encoded username
            return $this->sendCommand('Username', base64_encode($msg3), 235);
            break;
        case 'CRAM-MD5':
            // Start authentication
            if (!$this->sendCommand('AUTH CRAM-MD5', 'AUTH CRAM-MD5', 334)) {
                return false;
            }
            // Get the challenge
            $challenge = base64_decode(substr($this->last_reply, 4));

            // Build the response
            $response = $username . ' ' . $this->hmac($challenge, $password);

            // send encoded credentials
            return $this->sendCommand('Username', base64_encode($response), 235);
            break;
    }
    return true;
}

/**
 * Calculate an MD5 HMAC hash.
 * Works like hash_hmac('md5', $data, $key)
 * in case that function is not available
 * @param string $data The data to hash
 * @param string $key  The key to hash with
 * @access protected
 * @return string
 */
protected function hmac($data, $key)
{
    if (function_exists('hash_hmac')) {
        return hash_hmac('md5', $data, $key);
    }

    // The following borrowed from
    // http://php.net/manual/en/function.mhash.php#27225

    // RFC 2104 HMAC implementation for php.
    // Creates an md5 HMAC.
    // Eliminates the need to install mhash to compute a HMAC
    // Hacked by Lance Rushing

    $b = 64; // byte length for md5
    if (strlen($key) > $b) {
        $key = pack('H*', md5($key));
    }
    $key = str_pad($key, $b, chr(0x00));
    $ipad = str_pad('', $b, chr(0x36));
    $opad = str_pad('', $b, chr(0x5c));
    $k_ipad = $key ^ $ipad;
    $k_opad = $key ^ $opad;

    return md5($k_opad . pack('H*', md5($k_ipad . $data)));
}

/**
 * Check connection state.
 * @access public
 * @return bool True if connected.
 */
public function connected()
{
    if (!empty($this->smtp_conn)) {
        $sock_status = stream_get_meta_data($this->smtp_conn);
        if ($sock_status['eof']) {
            // the socket is valid but we are not connected
            if ($this->do_debug >= 1) {
                $this->edebug(
                    'SMTP NOTICE: EOF caught while checking if connected'
                );
            }
            $this->close();
            return false;
        }
        return true; // everything looks good
    }
    return false;
}

/**
 * Close the socket and clean up the state of the class.
 * Don't use this function without first trying to use QUIT.
 * @see quit()
 * @access public
 * @return void
 */
public function close()
{
    $this->error = null; // so there is no confusion
    $this->helo_rply = null;
    if (!empty($this->smtp_conn)) {
        // close the connection and cleanup
        fclose($this->smtp_conn);
        if ($this->do_debug >= 3) {
            $this->edebug('Connection: closed');
        }
        $this->smtp_conn = 0;
    }
}

/**
 * Send an SMTP DATA command.
 * Issues a data command and sends the msg_data to the server,
 * finializing the mail transaction. $msg_data is the message
 * that is to be send with the headers. Each header needs to be
 * on a single line followed by a <CRLF> with the message headers
 * and the message body being separated by and additional <CRLF>.
 * Implements rfc 821: DATA <CRLF>
 * @param string $msg_data Message data to send
 * @access public
 * @return bool
 */
public function data($msg_data)
{
```

```
if (!$this->sendCommand('DATA', 'DATA', 354)) {
    return false;
}

/* The server is ready to accept data!
 * according to rfc821 we should not send more than 1000
 * including the CRLF
 * characters on a single line so we will break the data up
 * into lines by \r and/or \n then if needed we will break
 * each of those into smaller lines to fit within the limit.
 * in addition we will be looking for lines that start with
 * a period '.' and append and additional period '.' to that
 * line. NOTE: this does not count towards limit.
 */

// Normalize the line breaks before exploding
$msg_data = str_replace("\r\n", "\n", $msg_data);
$msg_data = str_replace("\r", "\n", $msg_data);
$lines = explode("\n", $msg_data);

/* We need to find a good way to determine if headers are
 * in the msg_data or if it is a straight msg body
 * currently I am assuming rfc822 definitions of msg headers
 * and if the first field of the first line (':' separated)
 * does not contain a space then it _should_ be a header
 * and we can process all lines before a blank "" line as
 * headers.
 */

$field = substr($lines[0], 0, strpos($lines[0], ':'));
$in_headers = false;
if (!empty($field) && !strstr($field, ' ')) {
    $in_headers = true;
}

//RFC 2822 section 2.1.1 limit
$max_line_length = 998;

foreach ($lines as $line) {
    $lines_out = null;
    if ($line == '' && $in_headers) {
        $in_headers = false;
        }
        // ok we need to break this line up into several smaller lines
        while (strlen($line) > $max_line_length) {
            $pos = strrpos(substr($line, 0, $max_line_length), ' ');

            // Patch to fix DOS attack
            if (!$pos) {
                $pos = $max_line_length - 1;
                $lines_out[] = substr($line, 0, $pos);
                $line = substr($line, $pos);
            } else {
                $lines_out[] = substr($line, 0, $pos);
                $line = substr($line, $pos + 1);
            }

            /* If processing headers add a LWSP-char to the front of new line
             * rfc822 on long msg headers
             */
            if ($in_headers) {
                $line = "\t" . $line;
            }
        }
        $lines_out[] = $line;

        // send the lines to the server
        while (list(, $line_out) = @each($lines_out)) {
            if (strlen($line_out) > 0) {
                if (substr($line_out, 0, 1) == '.') {
                    $line_out = '.' . $line_out;
                }
            }
            $this->client_send($line_out . self::CRLF);
        }
    }

    // Message data has been sent, complete the command
    return $this->sendCommand('DATA END', '.', 250);
}

/**
 * Send an SMTP HELO or EHLO command.
 * Used to identify the sending server to the receiving server.
 * This makes sure that client and server are in a known state.
 * Implements from RFC 821: HELO <SP> <domain> <CRLF>
 * and RFC 2821 EHLO.
 * @param string $host The host name or IP to connect to
 * @access public
 * @return bool
 */
public function hello($host = '')
{
    // Try extended hello first (RFC 2821)
    if (!$this->sendHello('EHLO', $host)) {
        if (!$this->sendHello('HELO', $host)) {
            return false;
        }
    }

    return true;
}

/**
 * Send an SMTP HELO or EHLO command.
 * Low-level implementation used by hello()
 * @see hello()
 * @param string $hello The HELO string
 * @param string $host  The hostname to say we are
 * @access protected
 * @return bool
 */
protected function sendHello($hello, $host)
{
    $noerror = $this->sendCommand($hello, $hello . ' ' . $host, 250);
    $this->helo_rply = $this->last_reply;
    return $noerror;
}

/**
 * Send an SMTP MAIL command.
 * Starts a mail transaction from the email address specified in
 * $from. Returns true if successful or false otherwise. If True
 * the mail transaction is started and then one or more recipient
 * commands may be called followed by a data command.
 * Implements rfc 821: MAIL <SP> FROM:<reverse-path> <CRLF>
```

```
     * @param string $from Source address of this message
     * @access public
     * @return bool
     */
    public function mail($from)
    {
        $useVerp = ($this->do_verp ? ' XVERP' : '');
        return $this->sendCommand(
            'MAIL FROM',
            'MAIL FROM:<' . $from . '>' . $useVerp,
            250
        );
    }

    /**
     * Send an SMTP QUIT command.
     * Closes the socket if there is no error or the $close_on_error argument is true.
     * Implements from rfc 821: QUIT <CRLF>
     * @param bool $close_on_error Should the connection close if an error occurs?
     * @access public
     * @return bool
     */
    public function quit($close_on_error = true)
    {
        $noerror = $this->sendCommand('QUIT', 'QUIT', 221);
        $e = $this->error; //Save any error
        if ($noerror or $close_on_error) {
            $this->close();
            $this->error = $e; //Restore any error from the quit command
        }
        return $noerror;
    }

    /**
     * Send an SMTP RCPT command.
     * Sets the TO argument to $to.
     * Returns true if the recipient was accepted false if it was rejected.
     * Implements from rfc 821: RCPT <SP> TO:<forward-path> <CRLF>
     * @param string $to The address the message is being sent to
     * @access public
     * @return bool
     */
```

```
public function recipient($to)
{
    return $this->sendCommand(
        'RCPT TO ',
        'RCPT TO:<' . $to . '>',
        array(250, 251)
    );
}

/**
 * Send an SMTP RSET command.
 * Abort any transaction that is currently in progress.
 * Implements rfc 821: RSET <CRLF>
 * @access public
 * @return bool True on success.
 */
public function reset()
{
    return $this->sendCommand('RSET', 'RSET', 250);
}

/**
 * Send a command to an SMTP server and check its return code.
 * @param string $command       The command name - not sent to the server
 * @param string $commandstring The actual command to send
 * @param int|array $expect     One or more expected integer success codes
 * @access protected
 * @return bool True on success.
 */
protected function sendCommand($command, $commandstring, $expect)
{
    if (!$this->connected()) {
        $this->error = array(
            "error" => "Called $command without being connected"
        );
        return false;
    }
    $this->client_send($commandstring . self::CRLF);

    $reply = $this->get_lines();
    $code = substr($reply, 0, 3);
    if ($this->do_debug >= 2) {
        $this->edebug('SERVER -> CLIENT: ' . $reply);
    }

    if (!in_array($code, (array)$expect)) {
        $this->last_reply = null;
        $this->error = array(
            "error" => "$command command failed",
            "smtp_code" => $code,
            "detail" => substr($reply, 4)
        );
        if ($this->do_debug >= 1) {
            $this->edebug(
                'SMTP ERROR: ' . $this->error['error'] . ': ' . $reply
            );
        }
        return false;
    }

    $this->last_reply = $reply;
    $this->error = null;
    return true;
}

/**
 * Send an SMTP SAML command.
 * Starts a mail transaction from the email address specified in $from.
 * Returns true if successful or false otherwise. If True
 * the mail transaction is started and then one or more recipient
 * commands may be called followed by a data command. This command
 * will send the message to the users terminal if they are logged
 * in and send them an email.
 * Implements rfc 821: SAML <SP> FROM:<reverse-path> <CRLF>
 * @param string $from The address the message is from
 * @access public
 * @return bool
 */
public function sendAndMail($from)
{
    return $this->sendCommand("SAML", "SAML FROM:$from", 250);
}
```

```
/**
 * Send an SMTP VRFY command.
 * @param string $name The name to verify
 * @access public
 * @return bool
 */
public function verify($name)
{
    return $this->sendCommand("VRFY", "VRFY $name", array(250, 251));
}

/**
 * Send an SMTP NOOP command.
 * Used to keep keep-alives alive, doesn't actually do anything
 * @access public
 * @return bool
 */
public function noop()
{
    return $this->sendCommand("NOOP", "NOOP", 250);
}

/**
 * Send an SMTP TURN command.
 * This is an optional command for SMTP that this class does not support.
 * This method is here to make the RFC821 Definition
 * complete for this class and __may__ be implemented in future
 * Implements from rfc 821: TURN <CRLF>
 * @access public
 * @return bool
 */
public function turn()
{
    $this->error = array(
        'error' => 'The SMTP TURN command is not implemented'
    );
    if ($this->do_debug >= 1) {
        $this->edebug('SMTP NOTICE: ' . $this->error['error']);
    }
    return false;
}
```

```
/**
 * Send raw data to the server.
 * @param string $data The data to send
 * @access public
 * @return int|bool The number of bytes sent to the server or FALSE on error
 */
public function client_send($data)
{
    if ($this->do_debug >= 1) {
        $this->edebug("CLIENT -> SERVER: $data");
    }
    return fwrite($this->smtp_conn, $data);
}

/**
 * Get the latest error.
 * @access public
 * @return array
 */
public function getError()
{
    return $this->error;
}

/**
 * Get the last reply from the server.
 * @access public
 * @return string
 */
public function getLastReply()
{
    return $this->last_reply;
}

/**
 * Read the SMTP server's response.
 * Either before eof or socket timeout occurs on the operation.
 * With SMTP we can tell if we have more lines to read if the
 * 4th character is '-' symbol. If it is a space then we don't
 * need to read anything else.
 * @access protected
 * @return string
 */
protected function get_lines()
{
    $data = '';
    $endtime = 0;
    // If the connection is bad, give up now
    if (!is_resource($this->smtp_conn)) {
        return $data;
    }
    stream_set_timeout($this->smtp_conn, $this->Timeout);
    if ($this->Timelimit > 0) {
        $endtime = time() + $this->Timelimit;
    }
    while (is_resource($this->smtp_conn) && !feof($this->smtp_conn)) {
        $str = @fgets($this->smtp_conn, 515);
        if ($this->do_debug >= 4) {
            $this->edebug("SMTP -> get_lines(): \$data was \"$data\"");
            $this->edebug("SMTP -> get_lines(): \$str is \"$str\"");
        }
        $data .= $str;
        if ($this->do_debug >= 4) {
            $this->edebug("SMTP -> get_lines(): \$data is \"$data\"");
        }
        // if 4th character is a space, we are done reading, break the loop
        if (substr($str, 3, 1) == ' ') {
            break;
        }
        // Timed-out? Log and break
        $info = stream_get_meta_data($this->smtp_conn);
        if ($info['timed_out']) {
            if ($this->do_debug >= 4) {
                $this->edebug(
                    'SMTP -> get_lines(): timed-out (' . $this->Timeout . ' sec)'
                );
            }
            break;
        }
        // Now check if reads took too long
        if ($endtime) {
            if (time() > $endtime) {
                if ($this->do_debug >= 4) {
                    $this->edebug(
                        'SMTP -> get_lines(): timelimit reached ('
                        . $this->Timelimit . ' sec)'
                    );
                }
                break;
            }
        }
    }
    return $data;
}

/**
 * Enable or disable VERP address generation.
 * @param bool $enabled
 */
public function setVerp($enabled = false)
{
    $this->do_verp = $enabled;
}

/**
 * Get VERP address generation mode.
 * @return bool
 */
public function getVerp()
{
    return $this->do_verp;
}

/**
 * Set debug output method.
 * @param string $method The function/method to use for debugging output.
 */
public function setDebugOutput($method = 'echo')
{
    $this->Debugoutput = $method;
}

/**
 * Get debug output method.
 * @return string
 */
```

```
    public function getDebugOutput()
    {
        return $this->Debugoutput;
    }

    /**
     * Set debug output level.
     * @param int $level
     */
    public function setDebugLevel($level = 0)
    {
        $this->do_debug = $level;
    }

    /**
     * Get debug output level.
     * @return int
     */
    public function getDebugLevel()
    {
        return $this->do_debug;
    }

    /**
     * Set SMTP timeout.
     * @param int $timeout
     */
    public function setTimeout($timeout = 0)
    {
        $this->Timeout = $timeout;
    }

    /**
     * Get SMTP timeout.
     * @return int
     */
    public function getTimeout()
    {
        return $this->Timeout;
    }
}
```

68:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\extras\class.html2text.php
```
<?php
/*************************************************************************
 *                                                                       *
 * Converts HTML to formatted plain text                                 *
 *                                                                       *
 * Portions Copyright (c) 2005-2007 Jon Abernathy <jon@chuggnutt.com>    *
 * This version from https://github.com/mtibben/html2text                *
 *                                                                       *
 * This script is free software; you can redistribute it and/or modify   *
 * it under the terms of the GNU General Public License as published by  *
 * the Free Software Foundation; either version 2 of the License, or     *
 * (at your option) any later version.                                   *
 *                                                                       *
 * The GNU General Public License can be found at                        *
 * http://www.gnu.org/copyleft/gpl.html.                                 *
 *                                                                       *
 * This script is distributed in the hope that it will be useful,        *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the          *
 * GNU General Public License for more details.                          *
 *                                                                       *
 *************************************************************************/


class html2text
{

    /**
     *  Contains the HTML content to convert.
     *
     *  @var string $html
     *  @access public
     */
    public $html;

    /**
     *  Contains the converted, formatted text.
     *
     *  @var string $text
     *  @access public
     */
    public $text;

    /**
     *  Maximum width of the formatted text, in columns.
     *
     *  Set this value to 0 (or less) to ignore word wrapping
     *  and not constrain text to a fixed-width column.
     *
     *  @var integer $width
     *  @access public
     */
    public $width = 70;

    /**
     *  List of preg* regular expression patterns to search for,
     *  used in conjunction with $replace.
     *
     *  @var array $search
     *  @access public
     *  @see $replace
     */
    public $search = array(
        "/\r/",                                  // Non-legal carriage return
        "/[\n\t]+/",                             // Newlines and tabs
        '/<head[^>]*>.*?<\/head>/i',             // <head>
        '/<script[^>]*>.*?<\/script>/i',         // <script>s -- which strip_tags supposedly has problems
with
        '/<style[^>]*>.*?<\/style>/i',           // <style>s -- which strip_tags supposedly has problems with
        '/<p[^>]*>/i',                           // <P>
        '/<br[^>]*>/i',                          // <br>
        '/<i[^>]*>(.*?)<\/i>/i',                 // <i>
        '/<em[^>]*>(.*?)<\/em>/i',               // <em>
        '/(<ul[^>]*>|<\/ul>)/i',                 // <ul> and </ul>
        '/(<ol[^>]*>|<\/ol>)/i',                 // <ol> and </ol>
        '/<li[^>]*>(.*?)<\/li>/i',               // <li> and </li>
        '/<li[^>]*>/i',                          // <li>
        '/<hr[^>]*>/i',                          // <hr>
        '/<div[^>]*>/i',                         // <div>
        '/(<table[^>]*>|<\/table>)/i',           // <table> and </table>
        '/(<tr[^>]*>|<\/tr>)/i',                 // <tr> and </tr>
        '/<td[^>]*>(.*?)<\/td>/i',               // <td> and </td>
        '/<span class="_html2text_ignore">.+?<\/span>/i'  // <span
```

```
class="_html2text_ignore">...</span>
    );

    /**
     *  List of pattern replacements corresponding to patterns searched.
     *
     *  @var array $replace
     *  @access public
     *  @see $search
     */
    public $replace = array(
        '',                                     // Non-legal carriage return
        ' ',                                    // Newlines and tabs
        '',                                     // <head>
        '',                                     // <script>s -- which strip_tags supposedly has problems with
        '',                                     // <style>s -- which strip_tags supposedly has problems with
        "\n\n",                                 // <P>
        "\n",                                   // <br>
        '_\\1_',                                // <i>
        '_\\1_',                                // <em>
        "\n\n",                                 // <ul> and </ul>
        "\n\n",                                 // <ol> and </ol>
        "\t* \\1\n",                            // <li> and </li>
        "\n\t* ",                               // <li>
        "\n-------------------------\n",        // <hr>
        "<div>\n",                              // <div>
        "\n\n",                                 // <table> and </table>
        "\n",                                   // <tr> and </tr>
        "\t\t\\1\n",                            // <td> and </td>
        ""                                      // <span class="_html2text_ignore">...</span>
    );

    /**
     *  List of preg* regular expression patterns to search for,
     *  used in conjunction with $ent_replace.
     *
     *  @var array $ent_search
     *  @access public
     *  @see $ent_replace
     */
    public $ent_search = array(
        '/&(nbsp|#160);/i',                     // Non-breaking space
    '/&(quot|rdquo|ldquo|#8220|#8221|#147|#148);/i',
    // Double quotes
    '/&(apos|rsquo|lsquo|#8216|#8217);/i',   // Single quotes
    '/&gt;/i',                               // Greater-than
    '/&lt;/i',                               // Less-than
    '/&(copy|#169);/i',                      // Copyright
    '/&(trade|#8482|#153);/i',               // Trademark
    '/&(reg|#174);/i',                       // Registered
    '/&(mdash|#151|#8212);/i',               // mdash
    '/&(ndash|minus|#8211|#8722);/i',        // ndash
    '/&(bull|#149|#8226);/i',                // Bullet
    '/&(pound|#163);/i',                     // Pound sign
    '/&(euro|#8364);/i',                     // Euro sign
    '/&(amp|#38);/i',                        // Ampersand: see _converter()
    '/[ ]{2,}/',                             // Runs of spaces, post-handling
);

/**
 *  List of pattern replacements corresponding to patterns searched.
 *
 *  @var array $ent_replace
 *  @access public
 *  @see $ent_search
 */
public $ent_replace = array(
    ' ',                                     // Non-breaking space
    '"',                                     // Double quotes
    "'",                                     // Single quotes
    '>',
    '<',
    '(c)',
    '(tm)',
    '(R)',
    '--',
    '-',
    '*',
    'Â£',
    'EUR',                                   // Euro sign. € ?
    '|+|amp|+|',                             // Ampersand: see _converter()
    ' ',                                     // Runs of spaces, post-handling
);
```

```
/**
 *  List of preg* regular expression patterns to search for
 *  and replace using callback function.
 *
 *  @var array $callback_search
 *  @access public
 */
public $callback_search = array(
    '/<(a) [^>]*href=("|\')([^"\']+)\2([^>]*)>(.*?)<\/a>/i', // <a href="">
    '/<(h)[123456]( [^>]*)?>(.*?)<\/h[123456]>/i',         // h1 - h6
    '/<(b)( [^>]*)?>(.*?)<\/b>/i',                         // <b>
    '/<(strong)( [^>]*)?>(.*?)<\/strong>/i',               // <strong>
    '/<(th)( [^>]*)?>(.*?)<\/th>/i',                       // <th> and </th>
);

/**
 *  List of preg* regular expression patterns to search for in PRE body,
 *  used in conjunction with $pre_replace.
 *
 *  @var array $pre_search
 *  @access public
 *  @see $pre_replace
 */
public $pre_search = array(
    "/\n/",
    "/\t/",
    '/ /',
    '/<pre[^>]*>/',
    '/<\/pre>/'
);

/**
 *  List of pattern replacements corresponding to patterns searched for PRE body.
 *
 *  @var array $pre_replace
 *  @access public
 *  @see $pre_search
 */
public $pre_replace = array(
    '<br>',
    '&nbsp;&nbsp;&nbsp;&nbsp;',
    '&nbsp;',
    '',
    ''
);

/**
 *  Contains a list of HTML tags to allow in the resulting text.
 *
 *  @var string $allowed_tags
 *  @access public
 *  @see set_allowed_tags()
 */
public $allowed_tags = '';

/**
 *  Contains the base URL that relative links should resolve to.
 *
 *  @var string $url
 *  @access public
 */
public $url;

/**
 *  Indicates whether content in the $html variable has been converted yet.
 *
 *  @var boolean $_converted
 *  @access private
 *  @see $html, $text
 */
private $_converted = false;

/**
 *  Contains URL addresses from links to be rendered in plain text.
 *
 *  @var array $_link_list
 *  @access private
 *  @see _build_link_list()
 */
private $_link_list = array();


/**
 *  Various configuration options (able to be set in the constructor)
 *
 *  @var array $_options
 *  @access private
 */
private $_options = array(

    // 'none'
    // 'inline' (show links inline)
    // 'nextline' (show links on the next line)
    // 'table' (if a table of link URLs should be listed after the text.
    'do_links' => 'inline',

    //  Maximum width of the formatted text, in columns.
    //  Set this value to 0 (or less) to ignore word wrapping
    //  and not constrain text to a fixed-width column.
    'width' => 70,
);


/**
 *  Constructor.
 *
 *  If the HTML source string (or file) is supplied, the class
 *  will instantiate with that source propagated, all that has
 *  to be done it to call get_text().
 *
 *  @param string $source HTML content
 *  @param boolean $from_file Indicates $source is a file to pull content from
 *  @param array $options Set configuration options
 *  @access public
 *  @return void
 */
public function __construct( $source = '', $from_file = false, $options = array() )
{
    $this->_options = array_merge($this->_options, $options);

    if ( !empty($source) ) {
        $this->set_html($source, $from_file);
    }

    $this->set_base_url();
}
```

```
/**
 *  Loads source HTML into memory, either from $source string or a file.
 *
 *  @param string $source HTML content
 *  @param boolean $from_file Indicates $source is a file to pull content from
 *  @access public
 *  @return void
 */
public function set_html( $source, $from_file = false )
{
    if ( $from_file && file_exists($source) ) {
        $this->html = file_get_contents($source);
    }
    else
        $this->html = $source;

    $this->_converted = false;
}

/**
 *  Returns the text, converted from HTML.
 *
 *  @access public
 *  @return string
 */
public function get_text()
{
    if ( !$this->_converted ) {
        $this->_convert();
    }

    return $this->text;
}

/**
 *  Prints the text, converted from HTML.
 *
 *  @access public
 *  @return void
 */
public function print_text()
```

```
{
    print $this->get_text();
}

/**
 *  Alias to print_text(), operates identically.
 *
 *  @access public
 *  @return void
 *  @see print_text()
 */
public function p()
{
    print $this->get_text();
}

/**
 *  Sets the allowed HTML tags to pass through to the resulting text.
 *
 *  Tags should be in the form "<p>", with no corresponding closing tag.
 *
 *  @access public
 *  @return void
 */
public function set_allowed_tags( $allowed_tags = '' )
{
    if ( !empty($allowed_tags) ) {
        $this->allowed_tags = $allowed_tags;
    }
}

/**
 *  Sets a base URL to handle relative links.
 *
 *  @access public
 *  @return void
 */
public function set_base_url( $url = '' )
{
    if ( empty($url) ) {
        if ( !empty($_SERVER['HTTP_HOST']) ) {
            $this->url = 'http://' . $_SERVER['HTTP_HOST'];
        } else {
            $this->url = '';
        }
    } else {
        // Strip any trailing slashes for consistency (relative
        // URLs may already start with a slash like "/file.html")
        if ( substr($url, -1) == '/' ) {
            $url = substr($url, 0, -1);
        }
        $this->url = $url;
    }
}

/**
 *  Workhorse function that does actual conversion (calls _converter() method).
 *
 *  @access private
 *  @return void
 */
private function _convert()
{
    // Variables used for building the link list
    $this->_link_list = array();

    $text = trim(stripslashes($this->html));

    // Convert HTML to TXT
    $this->_converter($text);

    // Add link list
    if (!empty($this->_link_list)) {
        $text .= "\n\nLinks:\n------\n";
        foreach ($this->_link_list as $idx => $url) {
            $text .= '[' . ($idx+1) . '] ' . $url . "\n";
        }
    }

    $this->text = $text;

    $this->_converted = true;
}
```

```
/**
 *  Workhorse function that does actual conversion.
 *
 *  First performs custom tag replacement specified by $search and
 *  $replace arrays. Then strips any remaining HTML tags, reduces whitespace
 *  and newlines to a readable format, and word wraps the text to
 *  $this->_options['width'] characters.
 *
 *  @param string Reference to HTML content string
 *
 *  @access private
 *  @return void
 */
private function _converter(&$text)
{
    // Convert <BLOCKQUOTE> (before PRE!)
    $this->_convert_blockquotes($text);

    // Convert <PRE>
    $this->_convert_pre($text);

    // Run our defined tags search-and-replace
    $text = preg_replace($this->search, $this->replace, $text);

    // Run our defined tags search-and-replace with callback
    $text = preg_replace_callback($this->callback_search, array($this, '_preg_callback'), $text);

    // Strip any other HTML tags
    $text = strip_tags($text, $this->allowed_tags);

    // Run our defined entities/characters search-and-replace
    $text = preg_replace($this->ent_search, $this->ent_replace, $text);

    // Replace known html entities
    $text = html_entity_decode($text, ENT_QUOTES);

    // Remove unknown/unhandled entities (this cannot be done in search-and-replace block)
    $text = preg_replace('/&([a-zA-Z0-9]{2,6}|#[0-9]{2,4});/', '', $text);

    // Convert "|+|amp|+|" into "&", need to be done after handling of unknown entities
    // This properly handles situation of "&amp;quot;" in input string
    $text = str_replace('|+|amp|+|', '&', $text);
    // Bring down number of empty lines to 2 max
    $text = preg_replace("/\n\s+\n/", "\n\n", $text);
    $text = preg_replace("/[\n]{3,}/", "\n\n", $text);

    // remove leading empty lines (can be produced by eg. P tag on the beginning)
    $text = ltrim($text, "\n");

    // Wrap the text to a readable format
    // for PHP versions >= 4.0.2. Default width is 75
    // If width is 0 or less, don't wrap the text.
    if ( $this->_options['width'] > 0 ) {
        $text = wordwrap($text, $this->_options['width']);
    }
}

/**
 *  Helper function called by preg_replace() on link replacement.
 *
 *  Maintains an internal list of links to be displayed at the end of the
 *  text, with numeric indices to the original point in the text they
 *  appeared. Also makes an effort at identifying and handling absolute
 *  and relative links.
 *
 *  @param string $link URL of the link
 *  @param string $display Part of the text to associate number with
 *  @access private
 *  @return string
 */
private function _build_link_list( $link, $display, $link_override = null)
{
    $link_method = ($link_override) ? $link_override : $this->_options['do_links'];
    if ($link_method == 'none')
        return $display;



    // Ignored link types
    if (preg_match('!^(javascript:|mailto:|#)!i', $link)) {
        return $display;
    }
    if (preg_match('!^([a-z][a-z0-9.+-]+:)!i', $link)) {
        $url = $link;
    }
    else {
        $url = $this->url;
        if (substr($link, 0, 1) != '/') {
            $url .= '/';
        }
        $url .= "$link";
    }

    if ($link_method == 'table')
    {
        if (($index = array_search($url, $this->_link_list)) === false) {
            $index = count($this->_link_list);
            $this->_link_list[] = $url;
        }

        return $display . ' [' . ($index+1) . ']';
    }
    elseif ($link_method == 'nextline')
    {
        return $display . "\n[" . $url . ']';
    }
    else // link_method defaults to inline
    {
        return $display . ' [' . $url . ']';
    }
}

/**
 *  Helper function for PRE body conversion.
 *
 *  @param string HTML content
 *  @access private
 */
private function _convert_pre(&$text)
{
    // get the content of PRE element
    while (preg_match('/<pre[^>]*>(.*)<\/pre>/ismU', $text, $matches)) {
        $this->pre_content = $matches[1];

        // Run our defined tags search-and-replace with callback
        $this->pre_content = preg_replace_callback($this->callback_search,
            array($this, '_preg_callback'), $this->pre_content);

        // convert the content
        $this->pre_content = sprintf('<div><br>%s<br></div>',
            preg_replace($this->pre_search, $this->pre_replace, $this->pre_content));
        // replace the content (use callback because content can contain $0 variable)
        $text = preg_replace_callback('/<pre[^>]*>.*<\/pre>/ismU',
            array($this, '_preg_pre_callback'), $text, 1);

        // free memory
        $this->pre_content = '';
    }
}

/**
 *  Helper function for BLOCKQUOTE body conversion.
 *
 *  @param string HTML content
 *  @access private
 */
private function _convert_blockquotes(&$text)
{
    if (preg_match_all('/<\/*blockquote[^>]*>/i', $text, $matches, PREG_OFFSET_CAPTURE)) {
        $level = 0;
        $diff = 0;
        $start = 0;
        $taglen = 0;
        foreach ($matches[0] as $m) {
            if ($m[0][0] == '<' && $m[0][1] == '/') {
                $level--;
                if ($level < 0) {
                    $level = 0; // malformed HTML: go to next blockquote
                }
                else if ($level > 0) {
                    // skip inner blockquote
                }
                else {
                    $end  = $m[1];
                    $len  = $end - $taglen - $start;
                    // Get blockquote content
                    $body = substr($text, $start + $taglen - $diff, $len);
                    // Set text width
                    $p_width = $this->_options['width'];
                    if ($this->_options['width'] > 0) $this->_options['width'] -= 2;
                    // Convert blockquote content
                    $body = trim($body);
                    $this->_converter($body);
                    // Add citation markers and create PRE block
                    $body = preg_replace('/((^|\n)>*)/', '\\1> ', trim($body));
                    $body = '<pre>' . htmlspecialchars($body) . '</pre>';
                    // Re-set text width
                    $this->_options['width'] = $p_width;
                    // Replace content
                    $text = substr($text, 0, $start - $diff)
                        . $body . substr($text, $end + strlen($m[0]) - $diff);

                    $diff = $len + $taglen + strlen($m[0]) - strlen($body);
                    unset($body);
                }
            }
            else {
                if ($level == 0) {
                    $start = $m[1];
                    $taglen = strlen($m[0]);
                }
                $level ++;
            }
        }
    }
}

/**
 *  Callback function for preg_replace_callback use.
 *
 *  @param  array PREG matches
 *  @return string
 */
private function _preg_callback($matches)
{
    switch (strtolower($matches[1])) {
        case 'b':
        case 'strong':
            return $this->_toupper($matches[3]);
            case 'th':
                return $this->_toupper("\t\t". $matches[3] ."\n");
            case 'h':
                return $this->_toupper("\n\n". $matches[3] ."\n\n");
            case 'a':
                // override the link method
                $link_override = null;
                if (preg_match("/_html2text_link_(\w+)/", $matches[4], $link_override_match))
                {
                    $link_override = $link_override_match[1];
                }
                // Remove spaces in URL (#1487805)
                $url = str_replace(' ', '', $matches[3]);
                return $this->_build_link_list($url, $matches[5], $link_override);
        }
    }

    /**
     *  Callback function for preg_replace_callback use in PRE content handler.
     *
     *  @param  array PREG matches
     *  @return string
     */
    private function _preg_pre_callback($matches)
    {
        return $this->pre_content;
    }

    /**
     * Strtoupper function with HTML tags and entities handling.
     *
     * @param string $str Text to convert
     * @return string Converted text
     */
    private function _toupper($str)
    {
        // string can containg HTML tags
        $chunks = preg_split('/(<[^>]*>)/', $str, null, PREG_SPLIT_NO_EMPTY |
PREG_SPLIT_DELIM_CAPTURE);

        // convert toupper only the text between HTML tags
        foreach ($chunks as $idx => $chunk) {
            if ($chunk[0] != '<') {
                $chunks[$idx] = $this->_strtoupper($chunk);
            }
        }

        return implode($chunks);
    }

    /**
     * Strtoupper multibyte wrapper function with HTML entities handling.
     *
     * @param string $str Text to convert
     * @return string Converted text
     */
    private function _strtoupper($str)
    {
        $str = html_entity_decode($str, ENT_COMPAT);

        if (function_exists('mb_strtoupper'))
            $str = mb_strtoupper($str);
        else
            $str = strtoupper($str);

        $str = htmlspecialchars($str, ENT_COMPAT);

        return $str;
    }
}

69:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\extras\EasyPeasyICS.php
<?php

/* ------------------------------------------------------------------------ */
/* EasyPeasyICS
/* ------------------------------------------------------------------------ */
/* Manuel Reinhard, manu@sprain.ch
/* Twitter: @sprain
/* Web: www.sprain.ch
/*
/* Built with inspiration by
/" http://stackoverflow.com/questions/1463480/how-can-i-use-php-to-dynamically-publish-an-ical-
file-to-be-read-by-google-calend/1464355#1464355
```

```
/* ----------------------------------------------------------------------- */
/* History:
/* 2010/12/17 - Manuel Reinhard - when it all started
/* ----------------------------------------------------------------------- */

class EasyPeasyICS {

protected $calendarName;
protected $events = array();


/**
 * Constructor
 * @param string $calendarName
 */
public function __construct($calendarName=""){
$this->calendarName = $calendarName;
}//function


/**
 * Add event to calendar
 * @param string $calendarName
 */
public function addEvent($start, $end, $summary="", $description="", $url=""){
$this->events[] = array(
"start" => $start,
"end"   => $end,
"summary" => $summary,
"description" => $description,
"url" => $url
);
}//function


public function render($output = true){

//start Variable
$ics = "";

//Add header
$ics .= "BEGIN:VCALENDAR
METHOD:PUBLISH
VERSION:2.0
X-WR-CALNAME:".$this->calendarName."
PRODID:-//hacksw/handcal//NONSGML v1.0//EN";

//Add events
foreach($this->events as $event){
$ics .= "
BEGIN:VEVENT
UID:". md5(uniqid(mt_rand(), true)) ."@EasyPeasyICS.php
DTSTAMP:" . gmdate('Ymd').'T'. gmdate('His') . "Z
DTSTART:".gmdate('Ymd', $event["start"])."T".gmdate('His', $event["start"])."Z
DTEND:".gmdate('Ymd', $event["end"])."T".gmdate('His', $event["end"])."Z
SUMMARY:".str_replace("\n", "\\n", $event['summary'])."
DESCRIPTION:".str_replace("\n", "\\n", $event['description'])."
URL;VALUE=URI:".$event['url']."
END:VEVENT";
}//foreach


//Footer
$ics .= "
END:VCALENDAR";


if ($output) {
//Output
header('Content-type: text/calendar; charset=utf-8');
header('Content-Disposition: inline; filename='.$this->calendarName.'.ics');
echo $ics;
} else {
return $ics;
}

}//function

}//class
```

70:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\extras\htmlfilter.php

```
<?php
/**
 * htmlfilter.inc
 * ---------------
 * This set of functions allows you to filter html in order to remove
 * any malicious tags from it. Useful in cases when you need to filter
 * user input for any cross-site-scripting attempts.
 *
 * Copyright (C) 2002-2004 by Duke University
 *
 * This library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as published by the Free Software Foundation; either
 * version 2.1 of the License, or (at your option) any later version.
 *
 * This library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with this library; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA
 * 02110-1301  USA
 *
 * @AuthorKonstantin Riabitsev <icon@linux.duke.edu>
 * @Version 1.1 ($Date: 2011-07-04 14:02:23 -0400 (Mon, 04 Jul 2011) $)
 */

/**
 * @Author  Jim Jagielski <jim@jaguNET.com / jimjag@gmail.com>
 */

/**
 * This function returns the final tag out of the tag name, an array
 * of attributes, and the type of the tag. This function is called by
 * tln_sanitize internally.
 *
 * @param  $tagname the name of the tag.
 * @param  $attary the array of attributes and their values
 * @param  $tagtype The type of the tag (see in comments).
 * @return a string with the final tag representation.
 */
function tln_tagprint($tagname, $attary, $tagtype){
$me = 'tln_tagprint';
```

```
if ($tagtype == 2){
$fulltag = '</' . $tagname . '>';
} else {
$fulltag = '<' . $tagname;
if (is_array($attary) && sizeof($attary)){
$atts = Array();
while (list($attname, $attvalue) = each($attary)){
array_push($atts, "$attname=$attvalue");
}
$fulltag .= ' ' . join(' ', $atts);
}
if ($tagtype == 3){
$fulltag .= ' /';
}
$fulltag .= '>';
}
return $fulltag;
}

/**
 * A small helper function to use with array_walk. Modifies a by-ref
 * value and makes it lowercase.
 *
 * @param  $val a value passed by-ref.
 * @returnvoid since it modifies a by-ref value.
 */
function tln_casenormalize(&$val){
$val = strtolower($val);
}

/**
 * This function skips any whitespace from the current position within
 * a string and to the next non-whitespace value.
 *
 * @param  $body   the string
 * @param  $offset the offset within the string where we should start
 *   looking for the next non-whitespace character.
 * @return   the location within the $body where the next
 *   non-whitespace char is located.
 */
function tln_skipspace($body, $offset){
$me = 'tln_skipspace';
```

```
preg_match('/^(\s*)/s', substr($body, $offset), $matches);
if (sizeof($matches[1])){
$count = strlen($matches[1]);
$offset += $count;
}
return $offset;
}

/**
 * This function looks for the next character within a string.It's
 * really just a glorified "strpos", except it catches the failures
 * nicely.
 *
 * @param  $body   The string to look for needle in.
 * @param  $offset Start looking from this position.
 * @param  $needle The character/string to look for.
 * @return   location of the next occurrence of the needle, or
 *   strlen($body) if needle wasn't found.
 */
function tln_findnxstr($body, $offset, $needle){
$me = 'tln_findnxstr';
$pos = strpos($body, $needle, $offset);
if ($pos === FALSE){
$pos = strlen($body);
}
return $pos;
}

/**
 * This function takes a PCRE-style regexp and tries to match it
 * within the string.
 *
 * @param  $body   The string to look for needle in.
 * @param  $offset Start looking from here.
 * @param  $reg   A PCRE-style regex to match.
 * @return   Returns a false if no matches found, or an array
 *   with the following members:
 *   - integer with the location of the match within $body
 *   - string with whatever content between offset and the match
 *   - string with whatever it is we matched
 */
function tln_findnxreg($body, $offset, $reg){
```

```
$me = 'tln_findnxreg';
$matches = Array();
$retarr = Array();
$preg_rule = '%^(.*?)(' . $reg . ')%s';
preg_match($preg_rule, substr($body, $offset), $matches);
if (!isset($matches[0])){
$retarr = false;
} else {
$retarr[0] = $offset + strlen($matches[1]);
$retarr[1] = $matches[1];
$retarr[2] = $matches[2];
}
return $retarr;
}

/**
 * This function looks for the next tag.
 *
 * @param  $body   String where to look for the next tag.
 * @param  $offset Start looking from here.
 * @return   false if no more tags exist in the body, or
 *   an array with the following members:
 *   - string with the name of the tag
 *   - array with attributes and their values
 *   - integer with tag type (1, 2, or 3)
 *   - integer where the tag starts (starting "<")
 *   - integer where the tag ends (ending ">")
 *   first three members will be false, if the tag is invalid.
 */
function tln_getnxtag($body, $offset){
$me = 'tln_getnxtag';
if ($offset > strlen($body)){
return false;
}
$lt = tln_findnxstr($body, $offset, '<');
if ($lt == strlen($body)){
return false;
}
/**
 * We are here:
 * blah blah <tag attribute="value">
 * \---------^
 */
$pos = tln_skipspace($body, $lt + 1);
if ($pos >= strlen($body)){
return Array(false, false, false, $lt, strlen($body));
}
/**
 * There are 3 kinds of tags:
 * 1. Opening tag, e.g.:
 *  <a href="blah">
 * 2. Closing tag, e.g.:
 *  </a>
 * 3. XHTML-style content-less tag, e.g.:
 *  <img src="blah"/>
 */
$tagtype = false;
switch (substr($body, $pos, 1)){
case '/':
$tagtype = 2;
$pos++;
break;
case '!':
/**
 * A comment or an SGML declaration.
 */
if (substr($body, $pos+1, 2) == '--'){
$gt = strpos($body, '-->', $pos);
if ($gt === false){
$gt = strlen($body);
} else {
$gt += 2;
}
return Array(false, false, false, $lt, $gt);
} else {
$gt = tln_findnxstr($body, $pos, '>');
return Array(false, false, false, $lt, $gt);
}
break;
default:
/**
 * Assume tagtype 1 for now. If it's type 3, we'll switch values
 * later.
 */
```

```
$tagtype = 1;
break;
}

$tag_start = $pos;
$tagname = '';
/**
 * Look for next [\W-_], which will indicate the end of the tag name.
 */
$regary = tln_findnxreg($body, $pos, '[^\w\-_]');
if ($regary == false){
return Array(false, false, false, $lt, strlen($body));
}
list($pos, $tagname, $match) = $regary;
$tagname = strtolower($tagname);

/**
 * $match can be either of these:
 * '>'indicating the end of the tag entirely.
 * '\s' indicating the end of the tag name.
 * '/'indicating that this is type-3 xhtml tag.
 *
 * Whatever else we find there indicates an invalid tag.
 */
switch ($match){
case '/':
/**
 * This is an xhtml-style tag with a closing / at the
 * end, like so: <img src="blah"/>. Check if it's followed
 * by the closing bracket. If not, then this tag is invalid
 */
if (substr($body, $pos, 2) == '/>'){
$pos++;
$tagtype = 3;
} else {
$gt = tln_findnxstr($body, $pos, '>');
$retary = Array(false, false, false, $lt, $gt);
return $retary;
}
case '>':
return Array($tagname, false, $tagtype, $lt, $pos);
break;
```

```
default:
/**
 * Check if it's whitespace
 */
if (preg_match('/\s/', $match)){
} else {
/**
 * This is an invalid tag! Look for the next closing ">".
 */
$gt = tln_findnxstr($body, $lt, '>');
return Array(false, false, false, $lt, $gt);
}
}

/**
 * At this point we're here:
 * <tagname attribute='blah'>
 * \-------^
 *
 * At this point we loop in order to find all attributes.
 */
$attname = '';
$atttype = false;
$attary = Array();

while ($pos <= strlen($body)){
$pos = tln_skipspace($body, $pos);
if ($pos == strlen($body)){
/**
 * Non-closed tag.
 */
return Array(false, false, false, $lt, $pos);
}
/**
 * See if we arrived at a ">" or "/>", which means that we reached
 * the end of the tag.
 */
$matches = Array();
preg_match('%^(\s*)(>|/>)%s', substr($body, $pos), $matches);
if (isset($matches[0]) && $matches[0]){
/**
 * Yep. So we did.
 */
$pos += strlen($matches[1]);
if ($matches[2] == '/>'){
$tagtype = 3;
$pos++;
}
return Array($tagname, $attary, $tagtype, $lt, $pos);
}

/**
 * There are several types of attributes, with optional
 * [:space:] between members.
 * Type 1:
 * attrname[:space:]=[:space:]'CDATA'
 * Type 2:
 * attrname[:space:]=[:space:]"CDATA"
 * Type 3:
 * attr[:space:]=[:space:]CDATA
 * Type 4:
 * attrname
 *
 * We leave types 1 and 2 the same, type 3 we check for
 * '"' and convert to "&quot" if needed, then wrap in
 * double quotes. Type 4 we convert into:
 * attrname="yes".
 */
$regary = tln_findnxreg($body, $pos, '[^\w\-_]');
if ($regary == false){
/**
 * Looks like body ended before the end of tag.
 */
return Array(false, false, false, $lt, strlen($body));
}
list($pos, $attname, $match) = $regary;
$attname = strtolower($attname);
/**
 * We arrived at the end of attribute name. Several things possible
 * here:
 * '>'means the end of the tag and this is attribute type 4
 * '/'if followed by '>' means the same thing as above
 * '\s' means a lot of things -- look what it's followed by.
 *anything else means the attribute is invalid.
 */
switch($match){
case '/':
/**
 * This is an xhtml-style tag with a closing / at the
 * end, like so: <img src="blah"/>. Check if it's followed
 * by the closing bracket. If not, then this tag is invalid
 */
if (substr($body, $pos, 2) == '/>'){
$pos++;
$tagtype = 3;
} else {
$gt = tln_findnxstr($body, $pos, '>');
$retary = Array(false, false, false, $lt, $gt);
return $retary;
}
case '>':
$attary{$attname} = '"yes"';
return Array($tagname, $attary, $tagtype, $lt, $pos);
break;
default:
/**
 * Skip whitespace and see what we arrive at.
 */
$pos = tln_skipspace($body, $pos);
$char = substr($body, $pos, 1);
/**
 * Two things are valid here:
 * '=' means this is attribute type 1 2 or 3.
 * \w means this was attribute type 4.
 * anything else we ignore and re-loop. End of tag and
 * invalid stuff will be caught by our checks at the beginning
 * of the loop.
 */
if ($char == '='){
$pos++;
$pos = tln_skipspace($body, $pos);
/**
 * Here are 3 possibilities:
 * "'"attribute type 1
 * '"'attribute type 2
 * everything else is the content of tag type 3
 */
$quot = substr($body, $pos, 1);
if ($quot == '\''){
$regary = tln_findnxreg($body, $pos+1, '\'');
if ($regary == false){
return Array(false, false, false, $lt, strlen($body));
}
list($pos, $attval, $match) = $regary;
$pos++;
$attary{$attname} = '\'' . $attval . '\'';
} else if ($quot == '"'){
$regary = tln_findnxreg($body, $pos+1, '\"');
if ($regary == false){
return Array(false, false, false, $lt, strlen($body));
}
list($pos, $attval, $match) = $regary;
$pos++;
$attary{$attname} = '"' . $attval . '"';
} else {
/**
 * These are hateful. Look for \s, or >.
 */
$regary = tln_findnxreg($body, $pos, '[\s>]');
if ($regary == false){
return Array(false, false, false, $lt, strlen($body));
}
list($pos, $attval, $match) = $regary;
/**
 * If it's ">" it will be caught at the top.
 */
$attval = preg_replace('/\"/s', '&quot;', $attval);
$attary{$attname} = '"' . $attval . '"';
}
} else if (preg_match('|[\w/>]|', $char)) {
/**
 * That was attribute type 4.
 */
$attary{$attname} = '"yes"';
} else {
/**
 * An illegal character. Find next '>' and return.
 */
```

```
$gt = tln_findnxstr($body, $pos, '>');
return Array(false, false, false, $lt, $gt);
}
}
}
/**
 * The fact that we got here indicates that the tag end was never
 * found. Return invalid tag indication so it gets stripped.
 */
return Array(false, false, false, $lt, strlen($body));
}

/**
 * Translates entities into literal values so they can be checked.
 *
 * @param $attvalue the by-ref value to check.
 * @param $regexthe regular expression to check against.
 * @param $hexwhether the entites are hexadecimal.
 * @returnTrue or False depending on whether there were matches.
 */
function tln_deent(&$attvalue, $regex, $hex=false){
$me = 'tln_deent';
$ret_match = false;
preg_match_all($regex, $attvalue, $matches);
if (is_array($matches) && sizeof($matches[0]) > 0){
$repl = Array();
for ($i = 0; $i < sizeof($matches[0]); $i++){
$numval = $matches[1][$i];
if ($hex){
$numval = hexdec($numval);
}
$repl{$matches[0][$i]} = chr($numval);
}
$attvalue = strtr($attvalue, $repl);
return true;
} else {
return false;
}
}

/**
 * This function checks attribute values for entity-encoded values
 * and returns them translated into 8-bit strings so we can run
 * checks on them.
 *
 * @param  $attvalue A string to run entity check against.
 * @return Nothing, modifies a reference value.
 */
function tln_defang(&$attvalue){
$me = 'tln_defang';
/**
 * Skip this if there aren't ampersands or backslashes.
 */
if (strpos($attvalue, '&') === false
&& strpos($attvalue, '\\') === false){
return;
}
$m = false;
do {
$m = false;
$m = $m || tln_deent($attvalue, '/\&#0*(\d+);*/s');
$m = $m || tln_deent($attvalue, '/\&#x0*((\d|[a-f])+);*/si', true);
$m = $m || tln_deent($attvalue, '/\\\\(\d+)/s', true);
} while ($m == true);
$attvalue = stripslashes($attvalue);
}

/**
 * Kill any tabs, newlines, or carriage returns. Our friends the
 * makers of the browser with 95% market value decided that it'd
 * be funny to make "java[tab]script" be just as good as "javascript".
 *
 * @param  attvalue The attribute value before extraneous spaces removed.
 * @return attvalue Nothing, modifies a reference value.
 */
function tln_unspace(&$attvalue){
$me = 'tln_unspace';
if (strcspn($attvalue, "\t\r\n\0 ") != strlen($attvalue)){
$attvalue = str_replace(Array("\t", "\r", "\n", "\0", " "),
Array('',  '',  '',  '',  ''), $attvalue);
}
}

/**
```

```
 * This function runs various checks against the attributes.
 *
 * @param  $tagnameString with the name of the tag.
 * @param  $attaryArray with all tag attributes.
 * @param  $rm_attnamesSee description for tln_sanitize
 * @param  $bad_attvalsSee description for tln_sanitize
 * @param  $add_attr_to_tag See description for tln_sanitize
 * @returnArray with modified attributes.
 */
function tln_fixatts($tagname,
 $attary,
 $rm_attnames,
 $bad_attvals,
 $add_attr_to_tag
 ){
$me = 'tln_fixatts';
while (list($attname, $attvalue) = each($attary)){
/**
 * See if this attribute should be removed.
 */
foreach ($rm_attnames as $matchtag=>$matchattrs){
if (preg_match($matchtag, $tagname)){
foreach ($matchattrs as $matchattr){
if (preg_match($matchattr, $attname)){
unset($attary{$attname});
continue;
}
}
}
}
/**
 * Remove any backslashes, entities, or extraneous whitespace.
 */
tln_defang($attvalue);
tln_unspace($attvalue);

/**
 * Now let's run checks on the attvalues.
 * I don't expect anyone to comprehend this. If you do,
 * get in touch with me so I can drive to where you live and
 * shake your hand personally. :)
 */
foreach ($bad_attvals as $matchtag=>$matchattrs){
if (preg_match($matchtag, $tagname)){
foreach ($matchattrs as $matchattr=>$valary){
if (preg_match($matchattr, $attname)){
/**
 * There are two arrays in valary.
 * First is matches.
 * Second one is replacements
 */
list($valmatch, $valrepl) = $valary;
$newvalue = preg_replace($valmatch,$valrepl,$attvalue);
if ($newvalue != $attvalue){
$attary{$attname} = $newvalue;
}
}
}
}
}
}
/**
 * See if we need to append any attributes to this tag.
 */
foreach ($add_attr_to_tag as $matchtag=>$addattary){
if (preg_match($matchtag, $tagname)){
$attary = array_merge($attary, $addattary);
}
}
return $attary;
}

/**
 *
 * @param $bodythe string with HTML you wish to filter
 * @param $tag_listsee description above
 * @param $rm_tags_with_content see description above
 * @param $self_closing_tagssee description above
 * @param $force_tag_closingsee description above
 * @param $rm_attnamessee description above
 * @param $bad_attvalssee description above
 * @param $add_attr_to_tagsee description above
 * @returntln_sanitized html safe to show on your pages.
 */
```

```
function tln_sanitize($body,
  $tag_list,
  $rm_tags_with_content,
  $self_closing_tags,
  $force_tag_closing,
  $rm_attnames,
  $bad_attvals,
  $add_attr_to_tag
  )
{
$me = 'tln_sanitize';
/**
 * Normalize rm_tags and rm_tags_with_content.
 */
$rm_tags = array_shift($tag_list);
@array_walk($tag_list, 'tln_casenormalize');
@array_walk($rm_tags_with_content, 'tln_casenormalize');
@array_walk($self_closing_tags, 'tln_casenormalize');
/**
 * See if tag_list is of tags to remove or tags to allow.
 * false  means remove these tags
 * true  means allow these tags
 */
$curpos = 0;
$open_tags = Array();
$trusted = "<!-- begin tln_sanitized html -->\n";
$skip_content = false;
/**
 * Take care of netscape's stupid javascript entities like
 * &{alert('boo')};
 */
$body = preg_replace('/&(\{.*?\};)/si', '&amp;\\1', $body);
while (($curtag = tln_getnxtag($body, $curpos)) != FALSE){
list($tagname, $attary, $tagtype, $lt, $gt) = $curtag;
$free_content = substr($body, $curpos, $lt - $curpos);
if ($skip_content == false){
$trusted .= $free_content;
} else {
}
if ($tagname != FALSE){
if ($tagtype == 2){
if ($skip_content == $tagname){
```

```
/**
 * Got to the end of tag we needed to remove.
 */
$tagname = false;
$skip_content = false;
} else {
if ($skip_content == false){
if (isset($open_tags{$tagname}) &&
$open_tags{$tagname} > 0){
$open_tags{$tagname}--;
} else {
$tagname = false;
}
} else {
}
}
} else {
/**
 * $rm_tags_with_content
 */
if ($skip_content == false){
/**
 * See if this is a self-closing type and change
 * tagtype appropriately.
 */
if ($tagtype == 1
&& in_array($tagname, $self_closing_tags)){
$tagtype = 3;
}
/**
 * See if we should skip this tag and any content
 * inside it.
 */
if ($tagtype == 1
&& in_array($tagname, $rm_tags_with_content)){
$skip_content = $tagname;
} else {
if (($rm_tags == false
 && in_array($tagname, $tag_list)) ||
($rm_tags == true
 && !in_array($tagname, $tag_list))){
$tagname = false;
} else {
if ($tagtype == 1){
if (isset($open_tags{$tagname})){
$open_tags{$tagname}++;
} else {
$open_tags{$tagname} = 1;
}
}
/**
 * This is where we run other checks.
 */
if (is_array($attary) && sizeof($attary) > 0){
$attary = tln_fixatts($tagname,
  $attary,
  $rm_attnames,
  $bad_attvals,
  $add_attr_to_tag);
}
}
}
} else {
}
}
if ($tagname != false && $skip_content == false){
$trusted .= tln_tagprint($tagname, $attary, $tagtype);
}
} else {
}
$curpos = $gt + 1;
}
$trusted .= substr($body, $curpos, strlen($body) - $curpos);
if ($force_tag_closing == true){
foreach ($open_tags as $tagname=>$opentimes){
while ($opentimes > 0){
$trusted .= '</' . $tagname . '>';
$opentimes--;
}
}
$trusted .= "\n";
}
$trusted .= "<!-- end tln_sanitized html -->\n";
return $trusted;
```

```
}

//
// Use the nifty htmlfilter library
//


function HTMLFilter($body, $trans_image_path, $block_external_images = false) {

$tag_list = Array(
false,
"object",
"meta",
"html",
"head",
"base",
"link",
"frame",
"iframe",
"plaintext",
"marquee"
);

$rm_tags_with_content = Array(
"script",
"applet",
"embed",
"title",
"frameset",
"xmp",
"xml"
);

$self_closing_tags =  Array(
"img",
"br",
"hr",
"input",
"outbind"
);

$force_tag_closing = true;
```

```
$rm_attnames = Array(
"/.*/" =>
Array(
// "/target/i",
"/^on.*/i",
"/^dynsrc/i",
"/^data.*/i",
"/^lowsrc.*/i"
)
);

$bad_attvals = Array(
"/.*/" =>
Array(
"/^src|background/i" =>
Array(
Array(
"/^([\'\"])\s*\S+script\s*:.*([\'\"])/si",
"/^([\'\"])\s*mocha\s*:*.*([\'\"])/si",
"/^([\'\"])\s*about\s*:.*([\'\"])/si"
),
Array(
"\\1$trans_image_path\\2",
"\\1$trans_image_path\\2",
"\\1$trans_image_path\\2",
"\\1$trans_image_path\\2"
)
),
"/^href|action/i" =>
Array(
Array(
"/^([\'\"])\s*\S+script\s*:.*([\'\"])/si",
"/^([\'\"])\s*mocha\s*:*.*([\'\"])/si",
"/^([\'\"])\s*about\s*:.*([\'\"])/si"
),
Array(
"\\1#\\1",
"\\1#\\1",
"\\1#\\1",
"\\1#\\1"
)
),
"/^style/i" =>
Array(
Array(
"/expression/i",
"/binding/i",
"/behaviou*r/i",
"/include-source/i",
"/position\s*:\s*absolute/i",
"/url\s*\(\s*([\'\"])\s*\S+script\s*:.*([\'\"])\s*\)/si",
"/url\s*\(\s*([\'\"])\s*mocha\s*:.*([\'\"])\s*\)/si",
"/url\s*\(\s*([\'\"])\s*about\s*:.*([\'\"])\s*\)/si",
"/(.*)\s*:\s*url\s*\(\s*([\'\"]*)\s*\S+script\s*:.*([\'\"]*)\s*\)/si"
),
Array(
"idiocy",
"idiocy",
"idiocy",
"idiocy",
"",
"url(\\1#\\1)",
"url(\\1#\\1)",
"url(\\1#\\1)",
"url(\\1#\\1)",
"url(\\1#\\1)",
"\\1:url(\\2#\\3)"
)
)
)
);

if ($block_external_images){
array_push($bad_attvals{'/.*/'}{'/^src|background/i'}[0],
'/^([\'\"])\s*https*:.*([\'\"])/si');
array_push($bad_attvals{'/.*/'}{'/^src|background/i'}[1],
"\\1$trans_image_path\\1");
array_push($bad_attvals{'/.*/'}{'/^style/i'}[0],
'/url\(([\'\"])\s*https*:.*([\'\"])\)/si');
array_push($bad_attvals{'/.*/'}{'/^style/i'}[1],
"url(\\1$trans_image_path\\1)");
}
```

```
$add_attr_to_tag = Array(
"/^a$/i" =>
Array('target'=>'"_blank"')
);

$trusted = tln_sanitize($body,
$tag_list,
$rm_tags_with_content,
$self_closing_tags,
$force_tag_closing,
$rm_attnames,
$bad_attvals,
$add_attr_to_tag
);
return $trusted;
}

?>
```

71:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\extras\ntlm_sasl_client.php

```
<?php
/*
 * ntlm_sasl_client.php
 *
 * @(#) $Id: ntlm_sasl_client.php,v 1.3 2004/11/17 08:00:37 mlemos Exp $
 *
 **
 ** Source: http://www.phpclasses.org/browse/file/7495.html
 ** License: BSD (http://www.phpclasses.org/package/1888-PHP-Single-API-for-standard-
authentication-mechanisms.html)
 ** Bundled with Permission
 **
 */

define("SASL_NTLM_STATE_START",             0);
define("SASL_NTLM_STATE_IDENTIFY_DOMAIN",   1);
define("SASL_NTLM_STATE_RESPOND_CHALLENGE", 2);
define("SASL_NTLM_STATE_DONE",              3);

class ntlm_sasl_client_class
{
var $credentials=array();
var $state=SASL_NTLM_STATE_START;

Function Initialize(&$client)
{
if(!function_exists($function="mcrypt_encrypt")
|| !function_exists($function="mhash"))
{
$extensions=array(
"mcrypt_encrypt"=>"mcrypt",
"mhash"=>"mhash"
);
$client->error="the extension ".$extensions[$function]." required by the NTLM SASL client class is
not available in this PHP configuration";
return(0);
}
return(1);
}

Function ASCIIToUnicode($ascii)
{
for($unicode="",$a=0;$a<strlen($ascii);$a++)
$unicode.=substr($ascii,$a,1).chr(0);
return($unicode);
}

Function TypeMsg1($domain,$workstation)
{
$domain_length=strlen($domain);
$workstation_length=strlen($workstation);
$workstation_offset=32;
$domain_offset=$workstation_offset+$workstation_length;
return(
"NTLMSSP\0".
"\x01\x00\x00\x00".
"\x07\x32\x00\x00".
pack("v",$domain_length).
pack("v",$domain_length).
pack("V",$domain_offset).
pack("v",$workstation_length).
pack("v",$workstation_length).
pack("V",$workstation_offset).
$workstation.
$domain
);
}

Function NTLMResponse($challenge,$password)
{
$unicode=$this->ASCIIToUnicode($password);
$md4=mhash(MHASH_MD4,$unicode);
$padded=$md4.str_repeat(chr(0),21-strlen($md4));
$iv_size=mcrypt_get_iv_size(MCRYPT_DES,MCRYPT_MODE_ECB);
$iv=mcrypt_create_iv($iv_size,MCRYPT_RAND);
for($response="",$third=0;$third<21;$third+=7)
{
for($packed="",$p=$third;$p<$third+7;$p++)
$packed.=str_pad(decbin(ord(substr($padded,$p,1))),8,"0",STR_PAD_LEFT);
for($key="",$p=0;$p<strlen($packed);$p+=7)
{
$s=substr($packed,$p,7);
$b=$s.((substr_count($s,"1") % 2) ? "0" : "1");
$key.=chr(bindec($b));
}
$ciphertext=mcrypt_encrypt(MCRYPT_DES,$key,$challenge,MCRYPT_MODE_ECB,$iv);
$response.=$ciphertext;
}
return $response;
}

Function TypeMsg3($ntlm_response,$user,$domain,$workstation)
{
$domain_unicode=$this->ASCIIToUnicode($domain);
$domain_length=strlen($domain_unicode);
$domain_offset=64;
$user_unicode=$this->ASCIIToUnicode($user);
$user_length=strlen($user_unicode);
$user_offset=$domain_offset+$domain_length;
$workstation_unicode=$this->ASCIIToUnicode($workstation);
$workstation_length=strlen($workstation_unicode);
$workstation_offset=$user_offset+$user_length;
$lm="";
$lm_length=strlen($lm);
$lm_offset=$workstation_offset+$workstation_length;
$ntlm=$ntlm_response;
```

```
$ntlm_length=strlen($ntlm);
$ntlm_offset=$lm_offset+$lm_length;
$session="";
$session_length=strlen($session);
$session_offset=$ntlm_offset+$ntlm_length;
return(
"NTLMSSP\0".
"\x03\x00\x00\x00".
pack("v",$lm_length).
pack("v",$lm_length).
pack("V",$lm_offset).
pack("v",$ntlm_length).
pack("v",$ntlm_length).
pack("V",$ntlm_offset).
pack("v",$domain_length).
pack("v",$domain_length).
pack("V",$domain_offset).
pack("v",$user_length).
pack("v",$user_length).
pack("V",$user_offset).
pack("v",$workstation_length).
pack("v",$workstation_length).
pack("V",$workstation_offset).
pack("v",$session_length).
pack("v",$session_length).
pack("V",$session_offset).
"\x01\x02\x00\x00".
$domain_unicode.
$user_unicode.
$workstation_unicode.
$lm.
$ntlm
);
}

Function Start(&$client, &$message, &$interactions)
{
if($this->state!=SASL_NTLM_STATE_START)
{
$client->error="NTLM authentication state is not at the start";
return(SASL_FAIL);
}
```

```
$this->credentials=array(
"user"=>"",
"password"=>"",
"realm"=>"",
"workstation"=>""
);
$defaults=array();
$status=$client->GetCredentials($this->credentials,$defaults,$interactions);
if($status==SASL_CONTINUE)
$this->state=SASL_NTLM_STATE_IDENTIFY_DOMAIN;
Unset($message);
return($status);
}

Function Step(&$client, $response, &$message, &$interactions)
{
switch($this->state)
{
case SASL_NTLM_STATE_IDENTIFY_DOMAIN:
$message=$this->TypeMsg1($this->credentials["realm"],$this->credentials["workstation"]);
$this->state=SASL_NTLM_STATE_RESPOND_CHALLENGE;
break;
case SASL_NTLM_STATE_RESPOND_CHALLENGE:
$ntlm_response=$this->NTLMResponse(substr($response,24,8),$this->credentials["password"]);
$message=$this->TypeMsg3($ntlm_response,$this->credentials["user"],$this-
>credentials["realm"],$this->credentials["workstation"]);
$this->state=SASL_NTLM_STATE_DONE;
break;
case SASL_NTLM_STATE_DONE:
$client->error="NTLM authentication was finished without success";
return(SASL_FAIL);
default:
$client->error="invalid NTLM authentication step state";
return(SASL_FAIL);
}
return(SASL_CONTINUE);
}
};

?>
```

72:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-ar.php

```
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* Arabic Version, UTF-8
* by : bahjat al mostafa <bahjat983@hotmail.com>
*/

$PHPMAILER_LANG['authenticate']         = 'SMTP Error:    .';
$PHPMAILER_LANG['connect_host']         = 'SMTP Error:     SMTP.';
$PHPMAILER_LANG['data_not_accepted']    = 'SMTP Error:     .';
//$PHPMAILER_LANG['empty_message']        = 'Message body empty';
$PHPMAILER_LANG['encoding']             = '  : ';
$PHPMAILER_LANG['execute']              = '   : ';
$PHPMAILER_LANG['file_access']          = '   : ';
$PHPMAILER_LANG['file_open']            = 'File Error:    : ';
$PHPMAILER_LANG['from_failed']          = '       : ';
$PHPMAILER_LANG['instantiate']          = '    .';
//$PHPMAILER_LANG['invalid_address']        = 'Not sending, email address is invalid: ';
$PHPMAILER_LANG['mailer_not_supported'] = ' mailer  .';
//$PHPMAILER_LANG['provide_address']      = 'You must provide at least one recipient email
address.';
$PHPMAILER_LANG['recipients_failed']    = 'SMTP Error:   ' .
                                          '     : ';
$PHPMAILER_LANG['signing']              = '  : ';
//$PHPMAILER_LANG['smtp_connect_failed']  = 'SMTP Connect() failed.';
//$PHPMAILER_LANG['smtp_error']           = 'SMTP server error: ';
//$PHPMAILER_LANG['variable_set']         = 'Cannot set or reset variable: ';

73:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-be.php
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* Belarusian Version by Aleksander Maksymiuk <info@setpro.pl>
*/

$PHPMAILER_LANG['authenticate']         = ' SMTP:  .';
$PHPMAILER_LANG['connect_host']         = ' SMTP:     SMTP-.';
$PHPMAILER_LANG['data_not_accepted']    = ' SMTP:  .';
$PHPMAILER_LANG['empty_message']        = ' .';
$PHPMAILER_LANG['encoding']             = '  : ';
$PHPMAILER_LANG['execute']              = '  : ';
$PHPMAILER_LANG['file_access']          = '   : ';
```

```
$PHPMAILER_LANG['file_open']           = '  : ';
$PHPMAILER_LANG['from_failed']         = '  : ';
$PHPMAILER_LANG['instantiate']         = '   mail().';
$PHPMAILER_LANG['invalid_address']      = '  ,  email : ';
$PHPMAILER_LANG['provide_address']      = ',  ,  email .';
$PHPMAILER_LANG['mailer_not_supported'] = ' -    .';
$PHPMAILER_LANG['recipients_failed']    = ' SMTP:  : ';
$PHPMAILER_LANG['signing']             = '  : ';
$PHPMAILER_LANG['smtp_connect_failed']  = '   SMTP-.';
$PHPMAILER_LANG['smtp_error']          = ' SMTP: ';
$PHPMAILER_LANG['variable_set']        = '     : ';

74:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-br.php
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* Brazilian Portuguese Version
* By Paulo Henrique Garcia - paulo@controllerweb.com.br
* Edited by Lucas Guimarães - lucas@lucasguimaraes.com
*/

$PHPMAILER_LANG['authenticate']        = 'Erro de SMTP: Não foi possível autenticar.';
$PHPMAILER_LANG['connect_host']        = 'Erro de SMTP: Não foi possível conectar com o
servidor SMTP.';
$PHPMAILER_LANG['data_not_accepted']    = 'Erro de SMTP: Dados rejeitados.';
$PHPMAILER_LANG['empty_message']       = 'Corpo da mensagem vazio';
$PHPMAILER_LANG['encoding']            = 'Codificação desconhecida: ';
$PHPMAILER_LANG['execute']             = 'Não foi possível executar: ';
$PHPMAILER_LANG['file_access']         = 'Não foi possível acessar o arquivo: ';
$PHPMAILER_LANG['file_open']           = 'Erro de Arquivo: Não foi possível abrir o arquivo: ';
$PHPMAILER_LANG['from_failed']         = 'Os endereços dos rementententes a seguir falharam: ';
$PHPMAILER_LANG['instantiate']         = 'Não foi possível iniciar uma instância da função mail.';
$PHPMAILER_LANG['invalid_address']      = 'Não enviando, endereço de e-mail inválido: ';
$PHPMAILER_LANG['mailer_not_supported'] = ' mailer não é suportado.';
$PHPMAILER_LANG['provide_address']      = 'Você deve fornecer pelo menos um endereço de
destinatário de e-mail.';
$PHPMAILER_LANG['recipients_failed']    = 'Erro de SMTP: Os endereços de destinatário a
seguir falharam: ';
$PHPMAILER_LANG['signing']             = 'Erro ao assinar: ';
$PHPMAILER_LANG['smtp_connect_failed']  = 'SMTP Connect() falhou.';
$PHPMAILER_LANG['smtp_error']          = 'Erro de servidor SMTP: ';
$PHPMAILER_LANG['variable_set']        = 'Não foi possível definir ou resetar a variável: ';
```

75:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-ca.php

```
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* Catalan Version
* By Ivan: web AT microstudi DOT com
*/

$PHPMAILER_LANG['authenticate']         = 'Error SMTP: No s\'hapogut autenticar.';
$PHPMAILER_LANG['connect_host']         = 'Error SMTP: No es pot connectar al servidor SMTP.';
$PHPMAILER_LANG['data_not_accepted']    = 'Error SMTP: Dades no acceptades.';
//$PHPMAILER_LANG['empty_message']        = 'Message body empty';
$PHPMAILER_LANG['encoding']             = 'Codificació desconeguda: ';
$PHPMAILER_LANG['execute']              = 'No es pot executar: ';
$PHPMAILER_LANG['file_access']          = 'No es pot accedir a l\'arxiu: ';
$PHPMAILER_LANG['file_open']            = 'Error d\'Arxiu: No es pot obrir l\'arxiu: ';
$PHPMAILER_LANG['from_failed']          = 'La(s) següent(s) adreces de remitent han fallat: ';
$PHPMAILER_LANG['instantiate']          = 'No s\'ha pogut crear una instància de la funció Mail.';
//$PHPMAILER_LANG['invalid_address']        = 'Not sending, email address is invalid: ';
$PHPMAILER_LANG['mailer_not_supported'] = ' mailer no està suportat';
$PHPMAILER_LANG['provide_address']      = 'S\'ha de proveir almenys una adreça d\'email com
a destinatari.';
$PHPMAILER_LANG['recipients_failed']    = 'Error SMTP: Els següents destinataris han fallat: ';
//$PHPMAILER_LANG['signing']              = 'Signing Error: ';
//$PHPMAILER_LANG['smtp_connect_failed']  = 'SMTP Connect() failed.';
//$PHPMAILER_LANG['smtp_error']           = 'SMTP server error: ';
//$PHPMAILER_LANG['variable_set']         = 'Cannot set or reset variable: ';
```

76:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-ch.php

```
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* Chinese Version
* By LiuXin: www.80x86.cn/blog/
*/

$PHPMAILER_LANG['authenticate'] = 'SMTP ';
$PHPMAILER_LANG['connect_host'] = 'SMTP : SMTP';
$PHPMAILER_LANG['data_not_accepted'] = 'SMTP : ';
//$PHPMAILER_LANG['empty_message']        = 'Message body empty';
$PHPMAILER_LANG['encoding'] = '';
```

```
$PHPMAILER_LANG['execute'] = ': ';
$PHPMAILER_LANG['file_access'] = '';
$PHPMAILER_LANG['file_open'] = '';
$PHPMAILER_LANG['from_failed'] = ' ';
$PHPMAILER_LANG['instantiate'] = 'mail';
//$PHPMAILER_LANG['invalid_address']       = 'Not sending, email address is invalid: ';
$PHPMAILER_LANG['mailer_not_supported'] = ' ';
$PHPMAILER_LANG['provide_address'] = ' email';
$PHPMAILER_LANG['recipients_failed'] = 'SMTP    ';
//$PHPMAILER_LANG['signing']             = 'Signing Error: ';
//$PHPMAILER_LANG['smtp_connect_failed']  = 'SMTP Connect() failed.';
//$PHPMAILER_LANG['smtp_error']          = 'SMTP server error: ';
//$PHPMAILER_LANG['variable_set']        = 'Cannot set or reset variable: ';

77:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-cz.php
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* Czech Version
*/

$PHPMAILER_LANG['authenticate']         = 'Chyba SMTP: Autentizace selhala.';
$PHPMAILER_LANG['connect_host']         = 'Chyba SMTP: Nelze navázat spojení se SMTP
serverem.';
$PHPMAILER_LANG['data_not_accepted']    = 'Chyba SMTP: Data nebyla pijata.';
$PHPMAILER_LANG['empty_message']        = 'Prázdné tlo zprávy';
$PHPMAILER_LANG['encoding']             = 'Neznámé kódování: ';
$PHPMAILER_LANG['execute']              = 'Nelze provést: ';
$PHPMAILER_LANG['file_access']          = 'Nelze získat pístup k souboru: ';
$PHPMAILER_LANG['file_open']            = 'Chyba souboru: Nelze otevít soubor pro tení: ';
$PHPMAILER_LANG['from_failed']          = 'Následující adresa odesílatele je nesprávná: ';
$PHPMAILER_LANG['instantiate']          = 'Nelze vytvoit instanci emailové funkce.';
$PHPMAILER_LANG['invalid_address']      = 'Neplatná adresa: ';
$PHPMAILER_LANG['mailer_not_supported'] = ' mailer není podporován.';
$PHPMAILER_LANG['provide_address']      = 'Musíte zadat alespo jednu emailovou adresu
píjemce.';
$PHPMAILER_LANG['recipients_failed']    = 'Chyba SMTP: Následující adresy píjemc nejsou
správn: ';
$PHPMAILER_LANG['signing']              = 'Chyba pihlašování: ';
$PHPMAILER_LANG['smtp_connect_failed']  = 'SMTP Connect() selhal.';
$PHPMAILER_LANG['smtp_error']           = 'Chyba SMTP serveru: ';
$PHPMAILER_LANG['variable_set']         = 'Nelze nastavit nebo zmnit promnnou: ';
```

78:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-de.php

```php
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* German Version
*/

$PHPMAILER_LANG['authenticate']         = 'SMTP Fehler: Authentifizierung fehlgeschlagen.';
$PHPMAILER_LANG['connect_host']         = 'SMTP Fehler: Konnte keine Verbindung zum SMTP-
Host herstellen.';
$PHPMAILER_LANG['data_not_accepted']    = 'SMTP Fehler: Daten werden nicht akzeptiert.';
$PHPMAILER_LANG['empty_message']        = 'E-Mail Inhalt ist leer.';
$PHPMAILER_LANG['encoding']             = 'Unbekanntes Encoding-Format: ';
$PHPMAILER_LANG['execute']              = 'Konnte folgenden Befehl nicht ausführen: ';
$PHPMAILER_LANG['file_access']          = 'Zugriff auf folgende Datei fehlgeschlagen: ';
$PHPMAILER_LANG['file_open']            = 'Datei Fehler: konnte folgende Datei nicht öffnen: ';
$PHPMAILER_LANG['from_failed']          = 'Die folgende Absenderadresse ist nicht korrekt: ';
$PHPMAILER_LANG['instantiate']          = 'Mail Funktion konnte nicht initialisiert werden.';
$PHPMAILER_LANG['invalid_address']        = 'E-Mail wird nicht gesendet, die Adresse ist
ungültig.';
$PHPMAILER_LANG['mailer_not_supported'] = ' mailer wird nicht unterstützt.';
$PHPMAILER_LANG['provide_address']      = 'Bitte geben Sie mindestens eine Empfänger E-
Mailadresse an.';
$PHPMAILER_LANG['recipients_failed']    = 'SMTP Fehler: Die folgenden Empfänger sind nicht
korrekt: ';
$PHPMAILER_LANG['signing']              = 'Fehler beim Signieren: ';
$PHPMAILER_LANG['smtp_connect_failed']  = 'Verbindung zu SMTP Server fehlgeschlagen.';
$PHPMAILER_LANG['smtp_error']           = 'Fehler vom SMTP Server: ';
$PHPMAILER_LANG['variable_set']         = 'Kann Variable nicht setzen oder zurücksetzen: ';
```

79:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-dk.php

```php
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* Danish Version
* Author: Mikael Stokkebro <info@stokkebro.dk>
*/

$PHPMAILER_LANG['authenticate']         = 'SMTP fejl: Kunne ikke logge på.';
$PHPMAILER_LANG['connect_host']         = 'SMTP fejl: Kunne ikke tilslutte SMTP serveren.';
$PHPMAILER_LANG['data_not_accepted']    = 'SMTP fejl: Data kunne ikke accepteres.';
```

```
//$PHPMAILER_LANG['empty_message']        = 'Message body empty';
$PHPMAILER_LANG['encoding']             = 'Ukendt encode-format: ';
$PHPMAILER_LANG['execute']              = 'Kunne ikke køre: ';
$PHPMAILER_LANG['file_access']          = 'Ingen adgang til fil: ';
$PHPMAILER_LANG['file_open']            = 'Fil fejl: Kunne ikke åbne filen: ';
$PHPMAILER_LANG['from_failed']          = 'Følgende afsenderadresse er forkert: ';
$PHPMAILER_LANG['instantiate']          = 'Kunne ikke initialisere email funktionen.';
//$PHPMAILER_LANG['invalid_address']        = 'Not sending, email address is invalid: ';
$PHPMAILER_LANG['mailer_not_supported'] = ' mailer understøttes ikke.';
$PHPMAILER_LANG['provide_address']      = 'Du skal indtaste mindst en modtagers
emailadresse.';
$PHPMAILER_LANG['recipients_failed']    = 'SMTP fejl: Følgende modtagere er forkerte: ';
//$PHPMAILER_LANG['signing']              = 'Signing Error: ';
//$PHPMAILER_LANG['smtp_connect_failed']  = 'SMTP Connect() failed.';
//$PHPMAILER_LANG['smtp_error']           = 'SMTP server error: ';
//$PHPMAILER_LANG['variable_set']         = 'Cannot set or reset variable: ';

80:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-eo.php
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* Esperanto version
*/

$PHPMAILER_LANG['authenticate']         = 'Eraro de servilo SMTP : atentigo malsukcesis.';
$PHPMAILER_LANG['connect_host']         = 'Eraro de servilo SMTP : konektado al servilo
malsukcesis.';
$PHPMAILER_LANG['data_not_accepted']    = 'Eraro de servilo SMTP : neustaj datumoj.';
$PHPMAILER_LANG['empty_message']        = 'Teksto de mesao mankas.';
$PHPMAILER_LANG['encoding']             = 'Nekonata kodoprezento: ';
$PHPMAILER_LANG['execute']              = 'Lani rulumadon ne eblis: ';
$PHPMAILER_LANG['file_access']          = 'Aliro al dosiero ne sukcesis: ';
$PHPMAILER_LANG['file_open']            = 'Eraro de dosiero: malfermo neeblas: ';
$PHPMAILER_LANG['from_failed']          = 'Jena adreso de sendinto malsukcesis: ';
$PHPMAILER_LANG['instantiate']          = 'Genero de retmesaa funkcio neeblis.';
$PHPMAILER_LANG['invalid_address']      = 'Retadreso ne validas: ';
$PHPMAILER_LANG['mailer_not_supported'] = ' mesailo ne subtenata.';
$PHPMAILER_LANG['provide_address']      = 'Vi devas tajpi almena unu recevontan retadreson.';
$PHPMAILER_LANG['recipients_failed']    = 'Eraro de servilo SMTP : la jenaj potrecivuloj kazis
eraron: ';
$PHPMAILER_LANG['signing']              = 'Eraro de subskribo: ';
$PHPMAILER_LANG['smtp_connect_failed']  = 'SMTP konektado malsukcesis.';
```

```
$PHPMAILER_LANG['smtp_error']          = 'Eraro de servilo SMTP : ';
$PHPMAILER_LANG['variable_set']        = 'Variablo ne pravalorizeblas a ne repravalorizeblas: ';

81:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-es.php
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* Spanish version
* Versión en español
* Edited by Matt Sturdy - matt.sturdy@gmail.com
*/

$PHPMAILER_LANG['authenticate']        = 'Error SMTP: No se pudo autentificar.';
$PHPMAILER_LANG['connect_host']        = 'Error SMTP: No se pudo conectar al servidor
SMTP.';
$PHPMAILER_LANG['data_not_accepted']    = 'Error SMTP: Datos no aceptados.';
$PHPMAILER_LANG['empty_message']       = 'Cuerpo del mensaje vacío';
$PHPMAILER_LANG['encoding']            = 'Codificación desconocida: ';
$PHPMAILER_LANG['execute']             = 'No se pudo ejecutar: ';
$PHPMAILER_LANG['file_access']         = 'No se pudo acceder al archivo: ';
$PHPMAILER_LANG['file_open']           = 'Error de Archivo: No se pudo abrir el archivo: ';
$PHPMAILER_LANG['from_failed']         = 'La(s) siguiente(s) direcciones de remitente fallaron: ';
$PHPMAILER_LANG['instantiate']         = 'No se pudo crear una instancia de la función Mail.';
$PHPMAILER_LANG['invalid_address']      = 'No se pudo enviar: dirección de email inválido: ';
$PHPMAILER_LANG['mailer_not_supported'] = ' mailer no está soportado.';
$PHPMAILER_LANG['provide_address']      = 'Debe proveer al menos una dirección de email
como destino.';
$PHPMAILER_LANG['recipients_failed']    = 'Error SMTP: Los siguientes destinos fallaron: ';
$PHPMAILER_LANG['signing']             = 'Error al firmar: ';
$PHPMAILER_LANG['smtp_connect_failed']  = 'SMTP Connect() se falló.';
$PHPMAILER_LANG['smtp_error']          = 'Error del servidor SMTP: ';
$PHPMAILER_LANG['variable_set']        = 'No se pudo ajustar o reajustar la variable: ';

82:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-et.php
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* Estonian Version
* By Indrek Päri
* Revised By Elan Ruusamäe <glen@delfi.ee>
*/
```

```
$PHPMAILER_LANG['authenticate']         = 'SMTP Viga: Autoriseerimise viga.';
$PHPMAILER_LANG['connect_host']         = 'SMTP Viga: Ei õnnestunud luua ühendust SMTP
serveriga.';
$PHPMAILER_LANG['data_not_accepted']    = 'SMTP Viga: Vigased andmed.';
$PHPMAILER_LANG['empty_message']        = 'Tühi kirja sisu';
$PHPMAILER_LANG["encoding"]             = 'Tundmatu kodeering: ';
$PHPMAILER_LANG['execute']              = 'Tegevus ebaõnnestus: ';
$PHPMAILER_LANG['file_access']          = 'Pole piisavalt õiguseid järgneva faili avamiseks: ';
$PHPMAILER_LANG['file_open']            = 'Faili Viga: Faili avamine ebaõnnestus: ';
$PHPMAILER_LANG['from_failed']          = 'Järgnev saatja e-posti aadress on vigane: ';
$PHPMAILER_LANG['instantiate']          = 'mail funktiooni käivitamine ebaõnnestus.';
$PHPMAILER_LANG['invalid_address']      = 'Saatmine peatatud, e-posti address vigane: ';
$PHPMAILER_LANG['provide_address']      = 'Te peate määrama vähemalt ühe saaja e-posti
aadressi.';
$PHPMAILER_LANG['mailer_not_supported'] = ' maileri tugi puudub.';
$PHPMAILER_LANG['recipients_failed']    = 'SMTP Viga: Järgnevate saajate e-posti aadressid on
vigased: ';
$PHPMAILER_LANG["signing"]              = 'Viga allkirjastamisel: ';
$PHPMAILER_LANG['smtp_connect_failed']  = 'SMTP Connect() ebaõnnestus.';
$PHPMAILER_LANG['smtp_error']           = 'SMTP serveri viga: ';
$PHPMAILER_LANG['variable_set']         = 'Ei õnnestunud määrata või lähtestada muutujat: ';

83:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-fa.php
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* Persian/Farsi Version, UTF-8
* By: Ali Jazayeri <jaza.ali@gmail.com>
*/

$PHPMAILER_LANG['authenticate']         = 'SMTP Error:      .';
$PHPMAILER_LANG['connect_host']         = 'SMTP Error:    SMTP  .';
$PHPMAILER_LANG['data_not_accepted']    = 'SMTP Error:    .';
$PHPMAILER_LANG['empty_message']        = '    .';
$PHPMAILER_LANG['encoding']             = ' : ';
$PHPMAILER_LANG['execute']              = '   : ';
$PHPMAILER_LANG['file_access']          = '     : ';
$PHPMAILER_LANG['file_open']            = 'File Error:     : ';
$PHPMAILER_LANG['from_failed']          = '   : ';
$PHPMAILER_LANG['instantiate']          = '     .';
$PHPMAILER_LANG['invalid_address']      = '   : ';
$PHPMAILER_LANG['mailer_not_supported'] = ' mailer   .';
```

```
$PHPMAILER_LANG['provide_address']      = '      .';
$PHPMAILER_LANG['recipients_failed']    = 'SMTP Error:        : ';
$PHPMAILER_LANG['signing']              = '  : ';
$PHPMAILER_LANG['smtp_connect_failed']  = '    SMTP.';
$PHPMAILER_LANG['smtp_error']           = '  SMTP Server: ';
$PHPMAILER_LANG['variable_set']         = '       : ';

84:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-fi.php
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* Finnish Version
* By Jyry Kuukanen
*/

$PHPMAILER_LANG['authenticate']         = 'SMTP-virhe: käyttäjätunnistus epäonnistui.';
$PHPMAILER_LANG['connect_host']         = 'SMTP-virhe: yhteys palvelimeen ei onnistu.';
$PHPMAILER_LANG['data_not_accepted']    = 'SMTP-virhe: data on virheellinen.';
//$PHPMAILER_LANG['empty_message']        = 'Message body empty';
$PHPMAILER_LANG['encoding']             = 'Tuntematon koodaustyyppi: ';
$PHPMAILER_LANG['execute']              = 'Suoritus epäonnistui: ';
$PHPMAILER_LANG['file_access']          = 'Seuraavaan tiedostoon ei ole oikeuksia: ';
$PHPMAILER_LANG['file_open']            = 'Tiedostovirhe: Ei voida avata tiedostoa: ';
$PHPMAILER_LANG['from_failed']          = 'Seuraava lähettäjän osoite on virheellinen: ';
$PHPMAILER_LANG['instantiate']          = 'mail-funktion luonti epäonnistui.';
//$PHPMAILER_LANG['invalid_address']        = 'Not sending, email address is invalid: ';
$PHPMAILER_LANG['mailer_not_supported'] = 'postivälitintyyppiä ei tueta.';
$PHPMAILER_LANG['provide_address']      = 'Aseta vähintään yksi vastaanottajan
sähk&ouml;postiosoite.';
$PHPMAILER_LANG['recipients_failed']    = 'SMTP-virhe: seuraava vastaanottaja osoite on
virheellinen.';
$PHPMAILER_LANG['encoding']             = 'Tuntematon koodaustyyppi: ';
//$PHPMAILER_LANG['signing']              = 'Signing Error: ';
//$PHPMAILER_LANG['smtp_connect_failed']  = 'SMTP Connect() failed.';
//$PHPMAILER_LANG['smtp_error']           = 'SMTP server error: ';
//$PHPMAILER_LANG['variable_set']         = 'Cannot set or reset variable: ';

85:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-fo.php
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* Faroese Version [language of the Faroe Islands, a Danish dominion]
* This file created: 11-06-2004
* Supplied by Dávur Sørensen [www.profo-webdesign.dk]
*/

$PHPMAILER_LANG['authenticate']         = 'SMTP feilur: Kundi ikki góðkenna.';
$PHPMAILER_LANG['connect_host']         = 'SMTP feilur: Kundi ikki knýta samband við SMTP
vert.';
$PHPMAILER_LANG['data_not_accepted']    = 'SMTP feilur: Data ikki góðkent.';
//$PHPMAILER_LANG['empty_message']        = 'Message body empty';
$PHPMAILER_LANG['encoding']             = 'Ókend encoding: ';
$PHPMAILER_LANG['execute']              = 'Kundi ikki útføra: ';
$PHPMAILER_LANG['file_access']          = 'Kundi ikki tilganga fílu: ';
$PHPMAILER_LANG['file_open']            = 'Fílu feilur: Kundi ikki opna fílu: ';
$PHPMAILER_LANG['from_failed']          = 'fylgjandi Frá/From adressa miseydnaðist: ';
$PHPMAILER_LANG['instantiate']          = 'Kuni ikki instantiera mail funktión.';
//$PHPMAILER_LANG['invalid_address']        = 'Not sending, email address is invalid: ';
$PHPMAILER_LANG['mailer_not_supported'] = ' er ikki supporterað.';
$PHPMAILER_LANG['provide_address']      = 'Tú skal uppgeva minst móttakara-emailadressu(r).';
$PHPMAILER_LANG['recipients_failed']    = 'SMTP Feilur: Fylgjandi móttakarar miseydnaðust: ';
//$PHPMAILER_LANG['signing']              = 'Signing Error: ';
//$PHPMAILER_LANG['smtp_connect_failed']  = 'SMTP Connect() failed.';
//$PHPMAILER_LANG['smtp_error']           = 'SMTP server error: ';
//$PHPMAILER_LANG['variable_set']         = 'Cannot set or reset variable: ';

86:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-fr.php
<?php
/**
 * PHPMailer language file: refer to English translation for definitive list
 * French Version
 * Some French punctuation requires a thin non-breaking space (U+202F) character before it,
 * for example before a colon or exclamation mark.
 * There is one of these characters between these quotes: ""
 * @link http://unicode.org/udhr/n/notes_fra.html
 *
 */

$PHPMAILER_LANG['authenticate']         = 'Erreur SMTP: échec de l\'authentification.';
$PHPMAILER_LANG['connect_host']         = 'Erreur SMTP: impossible de se connecter au serveur
SMTP.';
$PHPMAILER_LANG['data_not_accepted']    = 'Erreur SMTP: données incorrectes.';
$PHPMAILER_LANG['empty_message']        = 'Corps de message vide.';
$PHPMAILER_LANG['encoding']             = 'Encodage inconnu: ';
```

```
$PHPMAILER_LANG['execute']             = 'Impossible de lancer l\'exécution: ';
$PHPMAILER_LANG['file_access']          = 'Impossible d\'accéder au fichier: ';
$PHPMAILER_LANG['file_open']            = 'Erreur de fichier: ouverture impossible: ';
$PHPMAILER_LANG['from_failed']          = 'L\'adresse d\'expéditeur suivante a échouée: ';
$PHPMAILER_LANG['instantiate']          = 'Impossible d\'instancier la fonction mail.';
$PHPMAILER_LANG['invalid_address']      = 'L\'adresse courriel n\'est pas valide: ';
$PHPMAILER_LANG['mailer_not_supported'] = ' client de messagerie non supporté.';
$PHPMAILER_LANG['provide_address']      = 'Vous devez fournir au moins une adresse de
destinataire.';
$PHPMAILER_LANG['recipients_failed']    = 'Erreur SMTP: les destinataires suivants sont en
erreur: ';
$PHPMAILER_LANG['signing']              = 'Erreur de signature: ';
$PHPMAILER_LANG['smtp_connect_failed']  = 'Échec de la connexion SMTP.';
$PHPMAILER_LANG['smtp_error']           = 'Erreur du serveur SMTP: ';
$PHPMAILER_LANG['variable_set']         = 'Ne peut initialiser ou réinitialiser une variable: ';

87:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-gl.php
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* Galician version
* Versión en galego
* Edited by Donato Rouco - donatorouco@gmail.com
*/

$PHPMAILER_LANG['authenticate']         = 'Erro SMTP: Non puido ser autentificado.';
$PHPMAILER_LANG['connect_host']         = 'Erro SMTP: Non puido conectar co servidor SMTP.';
$PHPMAILER_LANG['data_not_accepted']    = 'Erro SMTP: Datos non aceptados.';
$PHPMAILER_LANG['empty_message']        = 'Corpo da mensaxe vacía';
$PHPMAILER_LANG['encoding']             = 'Codificación descoñecida: ';
$PHPMAILER_LANG['execute']              = 'Non puido ser executado: ';
$PHPMAILER_LANG['file_access']          = 'Nob puido acceder ó arquivo: ';
$PHPMAILER_LANG['file_open']            = 'Erro de Arquivo: No puido abrir o arquivo: ';
$PHPMAILER_LANG['from_failed']          = 'A(s) seguinte(s) dirección(s) de remitente(s) deron
erro: ';
$PHPMAILER_LANG['instantiate']          = 'Non puido crear unha instancia da función Mail.';
$PHPMAILER_LANG['invalid_address']      = 'Non puido envia-lo correo: dirección de email
inválida: ';
$PHPMAILER_LANG['mailer_not_supported'] = ' mailer non está soportado.';
$PHPMAILER_LANG['provide_address']      = 'Debe engadir polo menos unha dirección de email
coma destino.';
$PHPMAILER_LANG['recipients_failed']    = 'Erro SMTP: Os seguintes destinos fallaron: ';
```

```
$PHPMAILER_LANG['signing']             = 'Erro ó firmar: ';
$PHPMAILER_LANG['smtp_connect_failed']  = 'SMTP Connect() fallou.';
$PHPMAILER_LANG['smtp_error']          = 'Erro do servidor SMTP: ';
$PHPMAILER_LANG['variable_set']        = 'Non puidemos axustar ou reaxustar a variábel: ';

88:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-he.php
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* Hebrew Version, UTF-8
* by : Ronny Sherer <ronny@hoojima.com>
*/

$PHPMAILER_LANG['authenticate']        = ' SMTP:   .';
$PHPMAILER_LANG['connect_host']        = ' SMTP:     SMTP.';
$PHPMAILER_LANG['data_not_accepted']    = ' SMTP:   .';
$PHPMAILER_LANG['empty_message']       = '  ';
$PHPMAILER_LANG['invalid_address']      = ' ';
$PHPMAILER_LANG['encoding']            = '  : ';
$PHPMAILER_LANG['execute']             = '   : ';
$PHPMAILER_LANG['file_access']         = '   : ';
$PHPMAILER_LANG['file_open']           = ' :    : ';
$PHPMAILER_LANG['from_failed']         = '   : ';
$PHPMAILER_LANG['instantiate']         = '     .';
$PHPMAILER_LANG['mailer_not_supported'] = '  .';
$PHPMAILER_LANG['provide_address']      = '       .';
$PHPMAILER_LANG['recipients_failed']    = ' SMTP:   : ';
$PHPMAILER_LANG['signing']             = ' : ';
$PHPMAILER_LANG['smtp_connect_failed']  = 'SMTP Connect() failed.';
$PHPMAILER_LANG['smtp_error']          = '  SMTP: ';
$PHPMAILER_LANG['variable_set']        = '      : ';

89:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-hu.php
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* Hungarian Version by @dominicus-75
*/

$PHPMAILER_LANG['authenticate']        = 'SMTP hiba: az azonosítás sikertelen.';
$PHPMAILER_LANG['connect_host']        = 'SMTP hiba: nem lehet kapcsolódni az SMTP-
szerverhez.';
```

```
$PHPMAILER_LANG['data_not_accepted']    = 'SMTP hiba: adatok visszautasítva.';
$PHPMAILER_LANG['empty_message']        = 'Üres az üzenettörzs.';
$PHPMAILER_LANG['encoding']             = 'Ismeretlen kódolás: ';
$PHPMAILER_LANG['execute']              = 'Nem lehet végrehajtani: ';
$PHPMAILER_LANG['file_access']          = 'A következ fájl nem elérhet: ';
$PHPMAILER_LANG['file_open']            = 'Fájl hiba: a következ fájlt nem lehet megnyitni: ';
$PHPMAILER_LANG['from_failed']          = 'A feladóként megadott következ cím hibás: ';
$PHPMAILER_LANG['instantiate']          = 'A PHP mail() függvényt nem sikerült végrehajtani.';
$PHPMAILER_LANG['invalid_address']      = 'Érvénytelen cím: ';
$PHPMAILER_LANG['mailer_not_supported'] = ' a mailer-osztály nem támogatott.';
$PHPMAILER_LANG['provide_address']      = 'Legalább egy címzettet fel kell tüntetni.';
$PHPMAILER_LANG['recipients_failed']    = 'SMTP hiba: a címzettként megadott következ címek
hibásak: ';
$PHPMAILER_LANG['signing']              = 'Hibás aláírás: ';
$PHPMAILER_LANG['smtp_connect_failed']  = 'Hiba az SMTP-kapcsolatban.';
$PHPMAILER_LANG['smtp_error']           = 'SMTP-szerver hiba: ';
$PHPMAILER_LANG['variable_set']         = 'A következ változók beállítása nem sikerült: ';

90:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-it.php
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* Italian version
* @package PHPMailer
* @author Ilias Bartolini <brain79@inwind.it>, Stefano Sabatini <sabas88@gmail.com>
*/

$PHPMAILER_LANG['authenticate']         = 'SMTP Error: Impossibile autenticarsi.';
$PHPMAILER_LANG['connect_host']         = 'SMTP Error: Impossibile connettersi all\'host SMTP.';
$PHPMAILER_LANG['data_not_accepted']    = 'SMTP Error: Dati non accettati dal server.';
$PHPMAILER_LANG['empty_message']        = 'Il corpo del messaggio è vuoto';
$PHPMAILER_LANG['encoding']             = 'Codifica dei caratteri sconosciuta: ';
$PHPMAILER_LANG['execute']              = 'Impossibile eseguire l\'operazione: ';
$PHPMAILER_LANG['file_access']          = 'Impossibile accedere al file: ';
$PHPMAILER_LANG['file_open']            = 'File Error: Impossibile aprire il file: ';
$PHPMAILER_LANG['from_failed']          = 'I seguenti indirizzi mittenti hanno generato errore: ';
$PHPMAILER_LANG['instantiate']          = 'Impossibile istanziare la funzione mail';
$PHPMAILER_LANG['invalid_address']        = 'Impossibile inviare, l\'indirizzo email non è valido: ';
$PHPMAILER_LANG['provide_address']      = 'Deve essere fornito almeno un indirizzo ricevente';
$PHPMAILER_LANG['mailer_not_supported'] = 'Mailer non supportato';
$PHPMAILER_LANG['recipients_failed']    = 'SMTP Error: I seguenti indirizzi destinatari hanno
generato un errore: ';
```

```
$PHPMAILER_LANG['signing']              = 'Errore nella firma: ';
$PHPMAILER_LANG['smtp_connect_failed']  = 'SMTP Connect() fallita.';
$PHPMAILER_LANG['smtp_error']           = 'Errore del server SMTP: ';
$PHPMAILER_LANG['variable_set']         = 'Impossibile impostare o resettare la variabile: ';

91:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-ja.php
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* Japanese Version
* By Mitsuhiro Yoshida - http://mitstek.com/
* Modified by Yoshi Sakai - http://bluemooninc.jp/
*/

$PHPMAILER_LANG['authenticate'] = 'SMTP: ';
$PHPMAILER_LANG['connect_host'] = 'SMTP: SMTP';
$PHPMAILER_LANG['data_not_accepted'] = 'SMTP: ';
//$PHPMAILER_LANG['empty_message']        = 'Message body empty';
$PHPMAILER_LANG['encoding'] = ': ';
$PHPMAILER_LANG['execute'] = ': ';
$PHPMAILER_LANG['file_access'] = ': ';
$PHPMAILER_LANG['file_open'] = ': : ';
$PHPMAILER_LANG['from_failed'] = 'From: ';
$PHPMAILER_LANG['instantiate'] = '';
//$PHPMAILER_LANG['invalid_address']        = 'Not sending, email address is invalid: ';
$PHPMAILER_LANG['provide_address'] = '1 ';
$PHPMAILER_LANG['mailer_not_supported'] = ' ';
$PHPMAILER_LANG['recipients_failed'] = 'SMTP:  : ';
//$PHPMAILER_LANG['signing']              = 'Signing Error: ';
//$PHPMAILER_LANG['smtp_connect_failed']  = 'SMTP Connect() failed.';
//$PHPMAILER_LANG['smtp_error']           = 'SMTP server error: ';
//$PHPMAILER_LANG['variable_set']         = 'Cannot set or reset variable: ';

92:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-lt.php
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* Lithuanian version by Dainius Kaupaitis <dk@sum.lt>
*/

$PHPMAILER_LANG['authenticate']         = 'SMTP klaida: autentifikacija nepavyko.';
$PHPMAILER_LANG['connect_host']         = 'SMTP klaida: nepavyksta prisijungti prie SMTP
```

```
stoties.';
$PHPMAILER_LANG['data_not_accepted']    = 'SMTP klaida: duomenys nepriimti.';
$PHPMAILER_LANG['empty_message']        = 'Laiško turinys tušias';
$PHPMAILER_LANG['encoding']             = 'Neatpažinta koduot: ';
$PHPMAILER_LANG['execute']              = 'Nepavyko vykdyti komandos: ';
$PHPMAILER_LANG['file_access']          = 'Byla nepasiekiama: ';
$PHPMAILER_LANG['file_open']            = 'Bylos klaida: Nepavyksta atidaryti: ';
$PHPMAILER_LANG['from_failed']          = 'Neteisingas siuntjo adresas: ';
$PHPMAILER_LANG['instantiate']          = 'Nepavyko paleisti mail funkcijos.';
$PHPMAILER_LANG['invalid_address']      = 'Neteisingas adresas';
$PHPMAILER_LANG['mailer_not_supported'] = ' pašto stotis nepalaikoma.';
$PHPMAILER_LANG['provide_address']      = 'Nurodykite bent vien gavjo adres.';
$PHPMAILER_LANG['recipients_failed']    = 'SMTP klaida: nepavyko išsisti šiems gavjams: ';
$PHPMAILER_LANG['signing']              = 'Prisijungimo klaida: ';
$PHPMAILER_LANG['smtp_connect_failed']  = 'SMTP susijungimo klaida';
$PHPMAILER_LANG['smtp_error']           = 'SMTP stoties klaida: ';
$PHPMAILER_LANG['variable_set']         = 'Nepavyko priskirti reikšms kintamajam: ';

93:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-lv.php
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* Latvian version by Eduards M. <e@npd.lv>
*/

$PHPMAILER_LANG['authenticate']         = 'SMTP kda: Autorizcija neizdevs.';
$PHPMAILER_LANG['connect_host']         = 'SMTP Kda: Nevar izveidot savienojumu ar SMTP
serveri.';
$PHPMAILER_LANG['data_not_accepted']    = 'SMTP Kda: Nepieem informciju.';
$PHPMAILER_LANG['empty_message']        = 'Ziojuma teksts ir tukšs';
$PHPMAILER_LANG['encoding']             = 'Neatpazts kodjums: ';
$PHPMAILER_LANG['execute']              = 'Neizdevs izpildt komandu: ';
$PHPMAILER_LANG['file_access']          = 'Fails nav pieejams: ';
$PHPMAILER_LANG['file_open']            = 'Faila kda: Nevar atvrt failu: ';
$PHPMAILER_LANG['from_failed']          = 'Nepareiza sttja adrese: ';
$PHPMAILER_LANG['instantiate']          = 'Nevar palaist stšanas funkciju.';
$PHPMAILER_LANG['invalid_address']      = 'Nepareiza adrese';
$PHPMAILER_LANG['mailer_not_supported'] = ' sttjs netiek atbalstts.';
$PHPMAILER_LANG['provide_address']      = 'Ldzu, nordiet vismaz vienu adrestu.';
$PHPMAILER_LANG['recipients_failed']    = 'SMTP kda: neizdevs nostt šdiem samjiem: ';
$PHPMAILER_LANG['signing']              = 'Autorizcijas kda: ';
$PHPMAILER_LANG['smtp_connect_failed']  = 'SMTP savienojuma kda';
```

```
$PHPMAILER_LANG['smtp_error']          = 'SMTP servera kda: ';
$PHPMAILER_LANG['variable_set']        = 'Nevar pieširt maing vrtbu: ';

94:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-nl.php
<?php
/**
* PHPMailer language file: refer to class.phpmailer.php for definitive list.
* Dutch Version by Tuxion <team@tuxion.nl>
*/

$PHPMAILER_LANG['authenticate']        = 'SMTP-fout: authenticatie mislukt.';//SMTP Error:
Could not authenticate.
$PHPMAILER_LANG['connect_host']        = 'SMTP-fout: kon niet verbinden met SMTP-
host.';//SMTP Error: Could not connect to SMTP host.
$PHPMAILER_LANG['data_not_accepted']    = 'SMTP-fout: data niet geaccepteerd.';//SMTP
Error: Data not accepted.
$PHPMAILER_LANG['empty_message']       = 'Berichttekst is leeg';//Message body empty
$PHPMAILER_LANG['encoding']            = 'Onbekende codering: ';//Unknown encoding:
$PHPMAILER_LANG['execute']             = 'Kon niet uitvoeren: ';//Could not execute:
$PHPMAILER_LANG['file_access']         = 'Kreeg geen toegang tot bestand: ';//Could not access
file:
$PHPMAILER_LANG['file_open']           = 'Bestandsfout: kon bestand niet openen: ';//File Error:
Could not open file:
$PHPMAILER_LANG['from_failed']         = 'Het volgende afzendersadres is mislukt: ';//The
following From address failed:
$PHPMAILER_LANG['instantiate']         = 'Kon mailfunctie niet initialiseren.';//Could not instantiate
mail function.
$PHPMAILER_LANG['invalid_address']     = 'Ongeldig adres';//Invalid address
$PHPMAILER_LANG['mailer_not_supported'] = ' mailer wordt niet ondersteund.';// mailer is not
supported.
$PHPMAILER_LANG['provide_address']     = 'Er moet minstens één ontvanger worden
opgegeven.';//You must provide at least one recipient email address.
$PHPMAILER_LANG['recipients_failed']    = 'SMTP-fout: de volgende ontvangers zijn mislukt:
';//SMTP Error: The following recipients failed:
$PHPMAILER_LANG['signing']             = 'Signeerfout: ';//Signing Error:
$PHPMAILER_LANG['smtp_connect_failed']  = 'SMTP Verbinding mislukt.';
$PHPMAILER_LANG['smtp_error']          = 'SMTP-serverfout: ';//SMTP server error:
$PHPMAILER_LANG['variable_set']        = 'Kan de volgende variablen niet instellen of resetten:
';//Cannot set or reset variable:

95:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-no.php
<?php
```

```
/**
* PHPMailer language file: refer to English translation for definitive list
* Norwegian Version
*/

$PHPMAILER_LANG['authenticate']         = 'SMTP Feil: Kunne ikke authentisere.';
$PHPMAILER_LANG['connect_host']         = 'SMTP Feil: Kunne ikke koble til SMTP host.';
$PHPMAILER_LANG['data_not_accepted']    = 'SMTP Feil: Data ble ikke akseptert.';
$PHPMAILER_LANG['empty_message']        = 'Meldingsinnholdet er tomt';
$PHPMAILER_LANG['encoding']             = 'Ukjent tegnkoding: ';
$PHPMAILER_LANG['execute']              = 'Kunne ikke utføre: ';
$PHPMAILER_LANG['file_access']          = 'Får ikke tilgang til filen: ';
$PHPMAILER_LANG['file_open']            = 'Fil feil: Kunne ikke åpne filen: ';
$PHPMAILER_LANG['from_failed']          = 'Følgende avsenderadresse feilet: ';
$PHPMAILER_LANG['instantiate']          = 'Kunne ikke initialisere mailfunksjonen.';
$PHPMAILER_LANG['invalid_address']      = 'Meldingen ble ikke sendt, følgende adresse er
ugyldig: ';
$PHPMAILER_LANG['provide_address']      = 'Du må angi minst en mottakeradresse.';
$PHPMAILER_LANG['mailer_not_supported'] = ' mailer er ikke supportert.';
$PHPMAILER_LANG['recipients_failed']    = 'SMTP Feil: Følgende mottagere feilet: ';
$PHPMAILER_LANG['signing']              = 'Signeringsfeil: ';
$PHPMAILER_LANG['smtp_connect_failed']  = 'SMTP Connect() feilet.';
$PHPMAILER_LANG['smtp_error']           = 'SMTP-serverfeil: ';
$PHPMAILER_LANG['variable_set']         = 'Kan ikke sette eller resette variabelen: ';

96:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-pl.php
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* Polish Version
*/

$PHPMAILER_LANG['authenticate']         = 'Bd SMTP: Nie mona przeprowadzi autentykacji.';
$PHPMAILER_LANG['connect_host']         = 'Bd SMTP: Nie mona poczy si z wybranym hostem.';
$PHPMAILER_LANG['data_not_accepted']    = 'Bd SMTP: Dane nie zostay przyjte.';
$PHPMAILER_LANG['empty_message']        = 'Wiadomo jest pusta.';
$PHPMAILER_LANG['encoding']             = 'Nieznany sposób kodowania znaków: ';
$PHPMAILER_LANG['execute']              = 'Nie mona uruchomi: ';
$PHPMAILER_LANG['file_access']          = 'Brak dostpu do pliku: ';
$PHPMAILER_LANG['file_open']            = 'Nie mona otworzy pliku: ';
$PHPMAILER_LANG['from_failed']          = 'Nastpujcy adres Nadawcy jest nieprawidowy: ';
$PHPMAILER_LANG['instantiate']          = 'Nie mona wywoa funkcji mail(). Sprawd konfiguracj
```

```
serwera.';
$PHPMAILER_LANG['invalid_address']      = 'Nie mona wysa wiadomoci, nastpujcy adres
Odbiorcy jest nieprawidowy: ';
$PHPMAILER_LANG['provide_address']      = 'Naley poda prawidowy adres email Odbiorcy.';
$PHPMAILER_LANG['mailer_not_supported'] = 'Wybrana metoda wysyki wiadomoci nie jest
obsugiwana.';
$PHPMAILER_LANG['recipients_failed']    = 'Bd SMTP: Nastpujcy odbiorcy s nieprawidowi: ';
$PHPMAILER_LANG['signing']              = 'Bd podpisywania wiadomoci: ';
$PHPMAILER_LANG['smtp_connect_failed']  = 'SMTP Connect() zakoczone niepowodzeniem.';
$PHPMAILER_LANG['smtp_error']           = 'Bd SMTP: ';
$PHPMAILER_LANG['variable_set']         = 'Nie mona zmieni zmiennej: ';

97:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-pt.php
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* pt-PT
* Portuguese (European) Version 1.0
* By Jonadabe - jonadabe@hotmail.com
*/

$PHPMAILER_LANG['authenticate']         = 'Erro do SMTP: Não foi possível realizar a
autenticação.';
$PHPMAILER_LANG['connect_host']         = 'Erro do SMTP: Não foi possível realizar ligação com
o servidor SMTP.';
$PHPMAILER_LANG['data_not_accepted']    = 'Erro do SMTP: Os dados foram rejeitados.';
$PHPMAILER_LANG['empty_message']        = 'A mensagem no e-mail está vazia.';
$PHPMAILER_LANG['encoding']             = 'Codificação desconhecida: ';
$PHPMAILER_LANG['execute']              = 'Não foi possível executar: ';
$PHPMAILER_LANG['file_access']          = 'Não foi possível aceder o ficheiro: ';
$PHPMAILER_LANG['file_open']            = 'Abertura do ficheiro: Não foi possível abrir o ficheiro: ';
$PHPMAILER_LANG['from_failed']          = 'Ocorreram falhas nos endereços dos seguintes
rementententes: ';
$PHPMAILER_LANG['instantiate']          = 'Não foi possível iniciar uma instância da função mail.';
$PHPMAILER_LANG['invalid_address']      = 'Não foi enviado nenhum e-mail para o endereço de
e-mail inválido: ';
$PHPMAILER_LANG['mailer_not_supported'] = ' mailer não é suportado.';
$PHPMAILER_LANG['provide_address']      = 'Tem de fornecer pelo menos um endereço como
destinatário do e-mail.';
$PHPMAILER_LANG['recipients_failed']    = 'Erro do SMTP: O endereço do seguinte destinatário
falhou: ';
$PHPMAILER_LANG['signing']              = 'Erro ao assinar: ';
```

```
$PHPMAILER_LANG['smtp_connect_failed']  = 'SMTP Connect() falhou.';
$PHPMAILER_LANG['smtp_error']          = 'Erro de servidor SMTP: ';
$PHPMAILER_LANG['variable_set']        = 'Não foi possível definir ou redefinir a variável: ';

98:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-ro.php
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* Romanian Version
* @package PHPMailer
* @author Catalin Constantin <catalin@dazoot.ro>
*/

$PHPMAILER_LANG['authenticate']        = 'Eroare SMTP: Nu a functionat autentificarea.';
$PHPMAILER_LANG['connect_host']        = 'Eroare SMTP: Nu m-am putut conecta la adresa
SMTP.';
$PHPMAILER_LANG['data_not_accepted']    = 'Eroare SMTP: Continutul mailului nu a fost
acceptat.';
//$PHPMAILER_LANG['empty_message']        = 'Message body empty';
$PHPMAILER_LANG['encoding']            = 'Encodare necunoscuta: ';
$PHPMAILER_LANG['execute']             = 'Nu pot executa:  ';
$PHPMAILER_LANG['file_access']         = 'Nu pot accesa fisierul: ';
$PHPMAILER_LANG['file_open']           = 'Eroare de fisier: Nu pot deschide fisierul: ';
$PHPMAILER_LANG['from_failed']         = 'Urmatoarele adrese From au dat eroare: ';
$PHPMAILER_LANG['instantiate']         = 'Nu am putut instantia functia mail.';
//$PHPMAILER_LANG['invalid_address']        = 'Not sending, email address is invalid: ';
$PHPMAILER_LANG['mailer_not_supported'] = ' mailer nu este suportat.';
$PHPMAILER_LANG['provide_address']      = 'Trebuie sa adaugati cel putin un recipient (adresa
de mail).';
$PHPMAILER_LANG['recipients_failed']    = 'Eroare SMTP: Urmatoarele adrese de mail au dat
eroare: ';
//$PHPMAILER_LANG['signing']             = 'Signing Error: ';
//$PHPMAILER_LANG['smtp_connect_failed']  = 'SMTP Connect() failed.';
//$PHPMAILER_LANG['smtp_error']          = 'SMTP server error: ';
//$PHPMAILER_LANG['variable_set']        = 'Cannot set or reset variable: ';

99:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-ru.php
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* Russian Version by Alexey Chumakov <alex@chumakov.ru>
*/
```

```
$PHPMAILER_LANG['authenticate']         = ' SMTP:  .';
$PHPMAILER_LANG['connect_host']         = ' SMTP:      SMTP.';
$PHPMAILER_LANG['data_not_accepted']    = ' SMTP:   .';
$PHPMAILER_LANG['encoding']             = '  : ';
$PHPMAILER_LANG['execute']              = '  : ';
$PHPMAILER_LANG['file_access']          = '   : ';
$PHPMAILER_LANG['file_open']            = ' :    : ';
$PHPMAILER_LANG['from_failed']          = '  : ';
$PHPMAILER_LANG['instantiate']          = '   mail.';
$PHPMAILER_LANG['provide_address']      = ',      e-mail .';
$PHPMAILER_LANG['mailer_not_supported'] = ' -    .';
$PHPMAILER_LANG['recipients_failed']    = ' SMTP:       : ';
$PHPMAILER_LANG['empty_message']        = '  ';
$PHPMAILER_LANG['invalid_address']      = ' ,   email : ';
$PHPMAILER_LANG['signing']              = ' : ';
$PHPMAILER_LANG['smtp_connect_failed']  = '   SMTP-';
$PHPMAILER_LANG['smtp_error']           = ' SMTP-: ';
$PHPMAILER_LANG['variable_set']         = '    : ';

100:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-se.php
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* Swedish Version
* Author: Johan Linnér <johan@linner.biz>
*/

$PHPMAILER_LANG['authenticate']         = 'SMTP fel: Kunde inte autentisera.';
$PHPMAILER_LANG['connect_host']         = 'SMTP fel: Kunde inte ansluta till SMTP-server.';
$PHPMAILER_LANG['data_not_accepted']    = 'SMTP fel: Data accepterades inte.';
//$PHPMAILER_LANG['empty_message']        = 'Message body empty';
$PHPMAILER_LANG['encoding']             = 'Okänt encode-format: ';
$PHPMAILER_LANG['execute']              = 'Kunde inte köra: ';
$PHPMAILER_LANG['file_access']          = 'Ingen åtkomst till fil: ';
$PHPMAILER_LANG['file_open']            = 'Fil fel: Kunde inte öppna fil: ';
$PHPMAILER_LANG['from_failed']          = 'Följande avsändaradress är felaktig: ';
$PHPMAILER_LANG['instantiate']          = 'Kunde inte initiera e-postfunktion.';
//$PHPMAILER_LANG['invalid_address']        = 'Not sending, email address is invalid: ';
$PHPMAILER_LANG['provide_address']      = 'Du måste ange minst en mottagares e-
postadress.';
$PHPMAILER_LANG['mailer_not_supported'] = ' mailer stöds inte.';
```

```
$PHPMAILER_LANG['recipients_failed']    = 'SMTP fel: Följande mottagare är felaktig: ';
//$PHPMAILER_LANG['signing']             = 'Signing Error: ';
//$PHPMAILER_LANG['smtp_connect_failed']  = 'SMTP Connect() failed.';
//$PHPMAILER_LANG['smtp_error']           = 'SMTP server error: ';
//$PHPMAILER_LANG['variable_set']         = 'Cannot set or reset variable: ';
```

101:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-sk.php

```
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* Slovak Version
* Author: Michal Tinka <michaltinka@gmail.com>
*/

$PHPMAILER_LANG['authenticate']         = 'SMTP Error: Chyba autentifikácie.';
$PHPMAILER_LANG['connect_host']         = 'SMTP Error: Nebolo možné nadviaza spojenie so SMTP serverom.';
$PHPMAILER_LANG['data_not_accepted']    = 'SMTP Error: Dáta neboli prijaté';
$PHPMAILER_LANG['empty_message']        = 'Prázdne telo správy.';
$PHPMAILER_LANG['encoding']             = 'Neznáme kódovanie: ';
$PHPMAILER_LANG['execute']              = 'Nedá sa vykona: ';
$PHPMAILER_LANG['file_access']          = 'Súbor nebol nájdený: ';
$PHPMAILER_LANG['file_open']            = 'File Error: Súbor sa otvori pre ítanie: ';
$PHPMAILER_LANG['from_failed']          = 'Následujúca adresa From je nesprávna: ';
$PHPMAILER_LANG['instantiate']          = 'Nedá sa vytvori inštancia emailovej funkcie.';
$PHPMAILER_LANG['invalid_address']        = 'Neodoslané, emailová adresa je nesprávna: ';
$PHPMAILER_LANG['mailer_not_supported'] = ' emailový klient nieje podporovaný.';
$PHPMAILER_LANG['provide_address']      = 'Musíte zada aspo jednu emailovú adresu príjemcu.';
$PHPMAILER_LANG['recipients_failed']    = 'SMTP Error: Adresy príjemcov niesu správne ';
$PHPMAILER_LANG['signing']              = 'Chyba prihlasovania: ';
$PHPMAILER_LANG['smtp_connect_failed']  = 'SMTP Connect() zlyhalo.';
$PHPMAILER_LANG['smtp_error']           = 'SMTP chyba serveru: ';
$PHPMAILER_LANG['variable_set']         = 'Nemožno nastavi alebo resetova premennú: ';
```

102:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-tr.php

```
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* Turkish version
* Türkçe Versiyonu
* ÝZYAZILIM - Elçin Özel - Can Yýlmaz - Mehmet Benlioðlu
```

```
*/

$PHPMAILER_LANG['authenticate']         = 'SMTP Hatas: Dorulanamyor.';
$PHPMAILER_LANG['connect_host']         = 'SMTP Hatas: SMTP hosta balanlamyor.';
$PHPMAILER_LANG['data_not_accepted']    = 'SMTP Hatas: Veri kabul edilmedi.';
$PHPMAILER_LANG['empty_message']        = 'Mesaj içerii bo';
$PHPMAILER_LANG['encoding']             = 'Bilinmeyen ifreleme: ';
$PHPMAILER_LANG['execute']              = 'Çaltrlamyor: ';
$PHPMAILER_LANG['file_access']          = 'Dosyaya eriilemiyor: ';
$PHPMAILER_LANG['file_open']            = 'Dosya Hatas: Dosya açlamyor: ';
$PHPMAILER_LANG['from_failed']          = 'Baarsz olan gönderici adresi: ';
$PHPMAILER_LANG['instantiate']          = 'Örnek mail fonksiyonu oluturulamad.';
$PHPMAILER_LANG['invalid_address']      = 'Gönderilmedi, email adresi geçersiz: ';
$PHPMAILER_LANG['provide_address']      = 'En az bir tane mail adresi belirtmek zorundasnz
alcnn email adresi.';
$PHPMAILER_LANG['mailer_not_supported'] = ' mailler desteklenmemektedir.';
$PHPMAILER_LANG['recipients_failed']    = 'SMTP Hatas: alclara ulamad: ';
$PHPMAILER_LANG['signing']              = 'mzalama hatas: ';
$PHPMAILER_LANG['smtp_connect_failed']  = 'SMTP balant() baarsz.';
$PHPMAILER_LANG['smtp_error']           = 'SMTP sunucu hatas: ';
$PHPMAILER_LANG['variable_set']         = 'Ayarlanamyor yada sfrlanamyor: ';

103:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-uk.php
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* Ukrainian Version by Yuriy Rudyy <yrudyy@prs.net.ua>
*/

$PHPMAILER_LANG['authenticate']         = ' SMTP:  .';
$PHPMAILER_LANG['connect_host']         = ' SMTP:      SMTP.';
$PHPMAILER_LANG['data_not_accepted']    = ' SMTP:   .';
$PHPMAILER_LANG['encoding']             = '  : ';
$PHPMAILER_LANG['execute']              = '  : ';
$PHPMAILER_LANG['file_access']          = '   : ';
$PHPMAILER_LANG['file_open']            = '  :    : ';
$PHPMAILER_LANG['from_failed']          = '  : ';
$PHPMAILER_LANG['instantiate']          = '   mail.';
$PHPMAILER_LANG['provide_address']      = '-,      e-mail .';
$PHPMAILER_LANG['mailer_not_supported'] = ' -    .';
$PHPMAILER_LANG['recipients_failed']    = ' SMTP:     : ';
$PHPMAILER_LANG['empty_message']        = '  ';
```

```
$PHPMAILER_LANG['invalid_address']      = ' ,   email : ';
$PHPMAILER_LANG['signing']              = ' : ';
$PHPMAILER_LANG['smtp_connect_failed']  = '   SMTP-';
$PHPMAILER_LANG['smtp_error']           = ' SMTP-: ';
$PHPMAILER_LANG['variable_set']         = '    : ';

104:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-zh.php
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* Traditional Chinese Version
* @author liqwei <liqwei@liqwei.com>
*/

$PHPMAILER_LANG['authenticate'] = 'SMTP ';
$PHPMAILER_LANG['connect_host'] = 'SMTP  SMTP ';
$PHPMAILER_LANG['data_not_accepted'] = 'SMTP ';
//$PHPMAILER_LANG['empty_message']        = 'Message body empty';
$PHPMAILER_LANG['encoding'] = ': ';
$PHPMAILER_LANG['file_access'] = '';
$PHPMAILER_LANG['file_open'] = '';
$PHPMAILER_LANG['from_failed'] = '';
$PHPMAILER_LANG['execute'] = '';
$PHPMAILER_LANG['instantiate'] = '';
//$PHPMAILER_LANG['invalid_address']        = 'Not sending, email address is invalid: ';
$PHPMAILER_LANG['provide_address'] = '';
$PHPMAILER_LANG['mailer_not_supported'] = '';
$PHPMAILER_LANG['recipients_failed'] = 'SMTP ';
//$PHPMAILER_LANG['signing']              = 'Signing Error: ';
//$PHPMAILER_LANG['smtp_connect_failed']  = 'SMTP Connect() failed.';
//$PHPMAILER_LANG['smtp_error']           = 'SMTP server error: ';
//$PHPMAILER_LANG['variable_set']         = 'Cannot set or reset variable: ';

105:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\language\phpmailer.lang-zh_cn.php
<?php
/**
* PHPMailer language file: refer to English translation for definitive list
* Simplified Chinese Version
* @author liqwei <liqwei@liqwei.com>
* @author young  <masxy@foxmail.com> blogbinaryoung.com
*/
```

```
$PHPMAILER_LANG['authenticate'] = 'SMTP ';
$PHPMAILER_LANG['connect_host'] = 'SMTP  SMTP ';
$PHPMAILER_LANG['data_not_accepted'] = 'SMTP ';
$PHPMAILER_LANG['empty_message']        = '';
$PHPMAILER_LANG['encoding'] = ': ';
$PHPMAILER_LANG['execute'] = '';
$PHPMAILER_LANG['file_access'] = '';
$PHPMAILER_LANG['file_open'] = '';
$PHPMAILER_LANG['from_failed'] = '';
$PHPMAILER_LANG['instantiate'] = '';
$PHPMAILER_LANG['invalid_address']        = '';
$PHPMAILER_LANG['mailer_not_supported'] = '';
$PHPMAILER_LANG['provide_address'] = '';
$PHPMAILER_LANG['recipients_failed'] = 'SMTP ';
$PHPMAILER_LANG['signing']              = '';
$PHPMAILER_LANG['smtp_connect_failed']  = 'SMTP';
$PHPMAILER_LANG['smtp_error']           = 'SMTP: ';
$PHPMAILER_LANG['variable_set']         = '';

106:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\mail\PHPMailerAutoload.php
<?php
/**
 * PHPMailer SPL autoloader.
 * PHP Version 5.0.0
 * @package PHPMailer
 * @link https://github.com/PHPMailer/PHPMailer/
 * @author Marcus Bointon (coolbru) <phpmailer@synchromedia.co.uk>
 * @author Jim Jagielski (jimjag) <jimjag@gmail.com>
 * @author Andy Prevost (codeworxtech) <codeworxtech@users.sourceforge.net>
 * @author Brent R. Matzelle (original founder)
 * @copyright 2013 Marcus Bointon
 * @copyright 2010 - 2012 Jim Jagielski
 * @copyright 2004 - 2009 Andy Prevost
 * @license http://www.gnu.org/copyleft/lesser.html GNU Lesser General Public License
 * @note This program is distributed in the hope that it will be useful - WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.
 */

/**
 * PHPMailer SPL autoloader.
 * @param string $classname The name of the class to load
 */
function PHPMailerAutoload($classname)
{
    //Can't use __DIR__ as it's only in PHP 5.3+
    $filename =
dirname(__FILE__).DIRECTORY_SEPARATOR.'class.'.strtolower($classname).'.php';
    if (is_readable($filename)) {
        require $filename;
    }
}

if (version_compare(PHP_VERSION, '5.1.2', '>=')) {
    //SPL autoloading was introduced in PHP 5.1.2
    if (version_compare(PHP_VERSION, '5.3.0', '>=')) {
        spl_autoload_register('PHPMailerAutoload', true, true);
    } else {
        spl_autoload_register('PHPMailerAutoload');
    }
} else {
    //Fall back to traditional autoload for old PHP versions
    function __autoload($classname)
    {
        PHPMailerAutoload($classname);
    }
}

107:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\pay\example\back.php
<?php

//

require_once('../quickpay_service.php');

//
mt_srand(quickpay_service::make_seed());

// =REFUND  =CONSUME_VOID, PRE_AUTH
//  PRE_AUTH_VOID(), PRE_AUTH_COMPLETE(), PRE_AUTH_VOID_COMPLETE()
$param['transType']             = quickpay_conf::REFUND;

$param['origQid']               = '201110281442120195882'; //qid,
$param['orderAmount']          = 11000;       //
$param['orderNumber']          = date('YmdHis') . strval(mt_rand(100, 999)); //()
$param['orderTime']            = date('YmdHis');   //, YYYYmmhhddHHMMSS
$param['orderCurrency']        = quickpay_conf::CURRENCY_CNY;  //

$param['customerIp']           = $_SERVER['REMOTE_ADDR'];  //IP
$param['frontEndUrl']          = "";    //URL,
$param['backEndUrl']           = "http://www.example.com/sdk/utf8/back_notify.php";    //URL

//

//
$pay_service = new quickpay_service($param, quickpay_conf::BACK_PAY);
$ret = $pay_service->post();

//,
$response = new quickpay_service($ret, quickpay_conf::RESPONSE);
if ($response->get('respCode') != quickpay_service::RESP_SUCCESS) { //
    $err = sprintf("Error: %d => %s", $response->get('respCode'), $response->get('respMsg'));
    throw new Exception($err);
}

//
$arr_ret = $response->get_args();

echo "\n" . var_export($arr_ret, true); //

?>

108:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\pay\example\back_notify.php
<?php

require_once('../quickpay_service.php');

try {
    $response = new quickpay_service($_POST, quickpay_conf::RESPONSE);
    if ($response->get('respCode') != quickpay_service::RESP_SUCCESS) {
        $err = sprintf("Error: %d => %s", $response->get('respCode'), $response->get('respMsg'));
        throw new Exception($err);
    }

    $arr_ret = $response->get_args();
    //
    //qid/

    //
    file_put_contents('notify.txt', var_export($arr_ret, true));

}
catch(Exception $exp) {
    //
    file_put_contents('notify.txt', var_export($exp, true));
}

?>

109:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\pay\example\front.php
<?php

//

require_once('../quickpay_service.php');

//
mt_srand(quickpay_service::make_seed());

$param['transType']             = quickpay_conf::CONSUME;  //CONSUME or PRE_AUTH

$param['orderAmount']           = 11000;        //
$param['orderNumber']           = date('YmdHis') . strval(mt_rand(100, 999)); //
$param['orderTime']             = date('YmdHis');   //, YYYYmmhhddHHMMSS
$param['orderCurrency']         = quickpay_conf::CURRENCY_CNY;  //CURRENCY_CNY=>

$param['customerIp']            = $_SERVER['REMOTE_ADDR'];  //IP
$param['frontEndUrl']           = "http://www.example.com/sdk/utf8/front_notify.php";    //URL
$param['backEndUrl']            = "http://www.example.com/sdk/utf8/back_notify.php";    //URL

/*
   $param['commodityUrl']          = "http://www.example.com/product?name=";  //URL
   $param['commodityName']         = '';   //
   $param['commodityUnitPrice']    = 11000;        //
   $param['commodityQuantity']     = 1;            //
//*/
```

```
//

$pay_service = new quickpay_service($param, quickpay_conf::FRONT_PAY);
$html = $pay_service->create_html();

header("Content-Type: text/html; charset=" . quickpay_conf::$pay_params['charset']);
echo $html; //post

?>

110:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\pay\example\front_notify.php
<?php

require_once('../quickpay_service.php');

try {
    $response = new quickpay_service($_POST, quickpay_conf::RESPONSE);
    if ($response->get('respCode') != quickpay_service::RESP_SUCCESS) {
        $err = sprintf("Error: %d => %s", $response->get('respCode'), $response->get('respMsg'));
        throw new Exception($err);
    }
    $arr_ret = $response->get_args();
    //
    echo " {$arr_ret['orderNumber']} ";

}
catch(Exception $exp) {
    $str .= var_export($exp, true);
    die("error happend: " . $str);
}

?>

111:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\pay\example\query.php
<?php

//

require_once('../quickpay_service.php');

//*
```

```
$transType   = quickpay_conf::CONSUME;
$orderNumber = "20111108150703852";
$orderTime   = "20111108150703";
// */

//
$param['transType']     = $transType;   //
$param['orderNumber']   = $orderNumber; //
$param['orderTime']     = $orderTime;   //

//
$query  = new quickpay_service($param, quickpay_conf::QUERY);
$ret    = $query->post();

//
$response = new quickpay_service($ret, quickpay_conf::RESPONSE);

//
$arr_ret = $response->get_args();
echo "<pre>\n" .  var_export($arr_ret, true) . "</pre>";

$respCode = $response->get('respCode');
$queryResult = $response->get('queryResult');

if ($queryResult == quickpay_service::QUERY_FAIL)
{
   echo "[respCode:{$respCode}]!\n";
   //,
}
else if ($queryResult == quickpay_service::QUERY_INVALID) {
   //
   echo "!\n";
}
else if ($respCode == quickpay_service::RESP_SUCCESS
      && $queryResult == quickpay_service::QUERY_SUCCESS)
{
   echo "!\n";
   //,
}
else if ($respCode == quickpay_service::RESP_SUCCESS
      && $queryResult == quickpay_service::QUERY_WAIT)
{
    echo "!\n";
}
else
{
    //
    $err = sprintf("Error[respCode:%d]", $response->get('respCode'));
    throw new Exception($err);
}


?>

112:F:\git\coin\\okbase.net\PHPCoins_V1.0\lib\pay\quickpay_conf.php
<?php

/*
 * @file    quickpay_service.inc.php
 * @author  fengmin(felix021@gmail.com)
 * @date    2011-08-22
 * @version $Revision$
 *
 */

class quickpay_conf
{

    const VERIFY_HTTPS_CERT = false;

    static $timezone        = "Asia/Shanghai"; //
    static $sign_method     = "md5"; //md5 (2011-08-22)

    static $security_key    = "88888888"; //

    //
    static $pay_params  = array(
        'version'       => '1.0.0',
        'charset'       => 'UTF-8', //UTF-8, GBK
        'merId'         => '105550149170027', //
        'acqCode'       => '',  //
        'merCode'       => '',  //
        'merAbbr'       => '',
    );
```

```
//*
static $front_pay_url   = "http://58.246.226.99/UpopWeb/api/Pay.action";
static $back_pay_url    = "http://58.246.226.99/UpopWeb/api/BSPay.action";
static $query_url       = "http://58.246.226.99/UpopWeb/api/Query.action";
//*/

/*
static $front_pay_url   = "https://www.epay.lxdns.com/UpopWeb/api/Pay.action";
static $back_pay_url    = "https://www.epay.lxdns.com/UpopWeb/api/BSPay.action";
static $query_url       = "https://www.epay.lxdns.com/UpopWeb/api/Query.action";
//*/

/*
static $front_pay_url   = "https://unionpaysecure.com/api/Pay.action";
static $back_pay_url    = "https://besvr.unionpaysecure.com/api/BSPay.action";
static $query_url       = "https://query.unionpaysecure.com/api/Query.action";
//*/

const FRONT_PAY = 1;
const BACK_PAY  = 2;
const RESPONSE  = 3;
const QUERY     = 4;

const CONSUME                = "01";
const CONSUME_VOID           = "31";
const PRE_AUTH               = "02";
const PRE_AUTH_VOID          = "32";
const PRE_AUTH_COMPLETE      = "03";
const PRE_AUTH_VOID_COMPLETE = "33";
const REFUND                 = "04";
const REGISTRATION           = "71";

const CURRENCY_CNY      = "156";

//
static $pay_params_empty = array(
    "origQid"           => "",
    "acqCode"           => "",
    "merCode"           => "",
    "commodityUrl"      => "",
    "commodityName"     => "",
    "commodityUnitPrice"=> "",
    "commodityQuantity" => "",
    "commodityDiscount" => "",
    "transferFee"       => "",
    "customerName"      => "",
    "defaultPayType"    => "",
    "defaultBankNumber" => "",
    "transTimeout"      => "",
    "merReserved"       => "",
);

//
static $pay_params_check = array(
    "version",
    "charset",
    "transType",
    "origQid",
    "merId",
    "merAbbr",
    "acqCode",
    "merCode",
    "commodityUrl",
    "commodityName",
    "commodityUnitPrice",
    "commodityQuantity",
    "commodityDiscount",
    "transferFee",
    "orderNumber",
    "orderAmount",
    "orderCurrency",
    "orderTime",
    "customerIp",
    "customerName",
    "defaultPayType",
    "defaultBankNumber",
    "transTimeout",
    "frontEndUrl",
    "backEndUrl",
    "merReserved",
);

//
```

```
static $query_params_check = array(
    "version",
    "charset",
    "transType",
    "merId",
    "orderNumber",
    "orderTime",
    "merReserved",
);

//
static $mer_params_reserved = array(
//  NEW NAME            OLD NAME
    "cardNumber",       "pan",
    "cardPasswd",       "password",
    "credentialType",   "idType",
    "cardCvn2",         "cvn",
    "cardExpire",       "expire",
    "credentialNumber", "idNo",
    "credentialName",   "name",
    "phoneNumber",      "mobile",
    "merAbstract",

    //tdb only
    "orderTimeoutDate",
    "origOrderNumber",
    "origOrderTime",
);

static $notify_param_check = array(
    "version",
    "charset",
    "transType",
    "respCode",
    "respMsg",
    "respTime",
    "merId",
    "merAbbr",
    "orderNumber",
    "traceNumber",
    "traceTime",
    "qid",
        "orderAmount",
        "orderCurrency",
        "settleAmount",
        "settleCurrency",
        "settleDate",
        "exchangeRate",
        "exchangeDate",
        "cupReserved",
        "signMethod",
        "signature",
    );

    static $sign_ignore_params = array(
        "bank",
    );
}

?>
```