Zend_Service_OrangeApi

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

Zend Framework: Zend_Service_OrangeApi Component Proposal

Proposed Component Name	Zend_Service_OrangeApi
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Service_OrangeApi
Proposers	Julien FILLIARD Laurent ARTUSIO
Zend Liaison	TBD
Revision	0.1 - 01 February 2011. (wiki revision: 92)

Table of Contents

1. Overview
2. References
3. Component Requirements, Constraints, and Acceptance Criteria
4. Dependencies on Other Framework Components
5. Theory of Operation
6. Milestones / Tasks
7. Class Index
8. Use Cases
Define your proxy settings
Send an SMS
Send an MMS
Request a mobile owner to opt-in for geolocation
Get a mobile device geolocation
Make a phone call
Create a conference call
Add a caller to an existing conference call
Create an outgoing call from a conference to a callee
Make interactive voice conferences
Make interactive voice outgoing calls
Make interactive voice mute a participant
Make interactive voice release calls
Make WebPhone calls
Get details for a previous API request
9. Class Skeletons

1. Overview

Orange API http://api.orange.com brings telecom services to web developers and integrators, with the Core solutions (web services to be integrated in web sites and applications), the Easy solutions (ready-to-use simple services) and Custom solutions (adapted to your specific needs). The Core services integrated in the Zend framework, through Zend_Service_OrangeApi allow to easily :

- send and receive text messages, and multimedia messages.
- get the location of Orange mobile phones, for any device
- control calls between fixed phones, mobile phones, and flash-based web phones. Flash phones can be basic examples from our website, or specifically developed by yourself.
- control voice conferences
- build interactive vocal services. If you're not familiar with VXML (V for Voice), our website contains various examples and FAQs

In order to make such APIs easy to use for Zend developers, Orange supplies a core Zend component.

All API requests are authenticated with a unique API key. In order to get an API key, you need to register on the Orange API website http://api.orange.com. All API response are sent in xml with a common structure. Those xml responses contain a "status_code" and a "status_msg" to report how the call occurred. Those response also contain a specific "Call_id" which can be used for getting specific history calls. If something failed, two kind of errors can be displayed:

- A functionalException
- A technicalException

2. References

- the Orange API website http://api.orange.com, providing full description of APIs, code samples, FAQs, tutorials and forums.

3. Component Requirements, Constraints, and Acceptance Criteria

- This component wraps Orange APIs in a raw manner (i.e parameters names, methods names, returned values), and does not restrict in any way the original features of Orange Apis.
- This component does not include parameters validity check, it will be done on the API server side.
- This component tranforms server-side errors in thrown exceptions, with full error codes and error messages.
- This component allow subclassing, for any useful purpose.

4. Dependencies on Other Framework Components

- Zend_Http_Client_Adapter_Proxy
- Zend_Rest_Client
- Zend_Service_Exception

5. Theory of Operation

Every Orange API is accessed through a single class which inherits the common Zend_Service_OrangeApi_Client class.
Each API class is named after the original Orange API name.
Each API class instanciation is done by passing the mandatory Orange access key ("id" parameter) to the constructor.

6. Milestones / Tasks

- Milestone 1: [DONE] Set a common architecture for all APIs
- Milestone 2: [DONE] Working prototype checked into the incubator supporting use cases.
- Milestone 4: [DONE] Initial documentation exists.

7. Class Index

- Zend_Service_OrangeApi_ClientAbstract
- Zend_Service_OrangeApi_Exception
- Zend_Service_OrangeApi_Sms
- Zend_Service_OrangeApi_Mms
- Zend_Service_OrangeApi_Location
- Zend_Service_OrangeApi_ClickToCall
- Zend_Service_OrangeApi_ClickToConf
- Zend_Service_OrangeApi_InteractiveVoice
- Zend_Service_OrangeApi_WebPhone
- Zend_Service_OrangeApi_CallDetails

8. Use Cases

Here's uses cases of each orange API. You can find a description of response codes in responseCodes.php in the common folder.

UC-01

Define your proxy settings

Should you need to go through a proxy, you can use the setProxy method available for all API classes.

proxySettings

```
//Zend_Service_OrangeApi_Sms initialisation
  $sms = new Zend_Service_OrangeApi_Sms($accessKey);

  //Proxy Settings
  $sms->setProxy( 'proxy_host' ,
                  'proxy_port' ,
                  'proxy_login' ,
                  'proxy_password'
                );
   //call service
   $sms->sendSms( $from ,
                  $to ,
                  $content ,
                  $ack
                );
```

UC-02

Send an SMS

sendSms sample

```
<?php
require_once('Zend/Service/OrangeApi/Exception.php');
require_once('Zend/Service/OrangeApi/Sms.php');

  //mandatory data to test Sms Api
  $accessKey = "my_access_key";
  $to = "callee_phone_number";
  $content="sms_content";


  //Zend_Service_OrangeApi_Sms initialization
  $sms = new Zend_Service_OrangeApi_Sms( $accessKey );

  /**
   * @desc sendSMS service
   * @param to : destination mobile number in international format (+33... +44...)
   * @param content : text message, limited to 160 characters.
   * @param from (optional) : number from which the SMS is sent (20345,31244, ...)
   * @param ack (optional) : request a delivery acknowledge if "true", default is "false".
   * Get your message delivery status using the getCallDetails API
   *
   */
  $sms->sendSms( $to ,
                 $content ,
                 $from ,
                 $ack
               );

   //Response management
   if($sms->getResponseCode()==200){
     echo" CallId : ".$sms->getResponseCallId()."\n";
     echo"<span>Send Sms successfull</span>";
   }
   else {
     $exception = new Zend_Service_OrangeApi_Exception($sms->getResponseCode()." -
".$sms->getResponseMessage());
     $exception->DisplayFunctionalException();
   }
?>
```

UC-3

Send an MMS

sendMMS sample

```
<?php
require_once('Zend/Service/OrangeApi/Exception.php');
require_once("Zend/Service/OrangeApi/Mms.php");

  //mandatory data to test Mms Api
  $accessKey = "my_access_key";
  $to="callee_phone_number";
  $content="test_Mms";

  //Zend_Service_OrangeApi_Mms initialization
  $mms = new Zend_Service_OrangeApi_Mms($accessKey );
  /**
   * @desc send an MMS message to mobile phones (currently limited to Orange France)
   * @param $to : destination number in international format (ex. : 336XXXXXXXX for a French mobile)
   * @param $content : MMS body. Supported formats : UTF-8
   * @param $subject (optional): MMS subject (limited to 40 characters), supported formats: UTF-8
   * @param $image (optional): attached Image file, supported formats : JPG, GIF
   * @param $sound (optional): attached sound file, supported formats : WAV, MP3, AMR
   * @param $video (optional): attached Video file, supported formats : AVI, 3GP
   * @param $from  (optional) : the short code sending the MMS (currently limited to 20345)
   *
   * Tip : if you create HTML pages to upload content to be sent via MMS, be aware that IE uploads
   * images in progressive jpeg, that are not supported by the MMS.
   *
   */

   $mms->sendMms( $to,
                  $content,
                  $subject,
                  $image,
                  $sound,
               $video,
                  $from
                );

   //Response management
   if($mms->getResponseCode()==200){
     echo" CallId : ".$mms->getResponseCallId()."\n";
     echo"<span>Send Mms successfull</span>";
   }
   else {
     $exception = new Zend_Service_OrangeApi_Exception($mms->getResponseCode()." -
".$mms->getResponseMessage());
     $exception->DisplayFunctionalException();
   }
?>
```

UC-04

Request a mobile owner to opt-in for geolocation

Location authorization sample

```
<?php
require_once('Zend/Service/OrangeApi/Exception.php');
require_once('Zend/Service/OrangeApi/Location.php');

  //mandatory data to test geolocation authorization Api
  $accessKey = "my_access_key";
  $number= "phone_number_for_authorization";

  //Zend_Service_OrangeApi_Location initialization
  $location = new Zend_Service_OrangeApi_Location( $accessKey );

  /**
   * @desc send SMS invitation to the mobile in order to get opt-in for geolocation.
   * The owner has to reply via SMS to confirm opt-in.
   * Geolocation requests until then will be rejected.
   * @param number : mobile phone number to be further located (international format)
   */

  $location->createAuthorization( $number );

  //Response management
  if($location->getResponseCode()==200){
    echo" CallId : ".$location->getResponseCallId()."\n";
    echo"<span>createAuthorization successfull</span>";
  }
  else {
    $exception = new Zend_Service_OrangeApi_Exception($location->getResponseCode()." -
".$location->getResponseMessage());
    $exception->DisplayFunctionalException();
  }
?>
```

UC-05

Get a mobile device geolocation

Get Location sample

```
<?php
require_once('Zend/Service/OrangeApi/Exception.php');
require_once('Zend/Service/OrangeApi/Location.php');

  //mandatory data to test geolocation Api
  $accessKey = "my_access_key";
  $number = "phone_number_for_localisation";

  //Zend_Service_OrangeApi_Location initialization
  $location = new Zend_Service_OrangeApi_Location( $accessKey );

  /**
   * @desc get the location of a previously opt-ined mobile phone
   * @param number :  mobile phone number of the mobile to be located (international format).
   */

  $location->getLocation( $number );

  //Response management
  if($location->getResponseCode()==200){
  echo"<div id='result'>
   <span style='width:150px;'>Location :<br>

     X :".$location->getX()."<br />

     Y : ".$location->getY()."<br />

     Radius : ".$location->getRad()."

   </span>
  </div>";

  }
?>
```

UC-06

Make a phone call

Click-to-call sample

```
<?php
require_once('Zend/Service/OrangeApi/Exception.php');
require_once('Zend/Service/OrangeApi/ClickToCall.php');

  //mandatory data to test Click-to-call Api
  $accessKey = "my_access_key";
  $from = "caller_number";
  $to = "callee_number";

  //Zend_Service_OrangeApi_ClickToCall initialization
  $clickToCall = new Zend_Service_OrangeApi_ClickToCall( $accessKey );

 /**
  * @desc make a phone call between 2 phone devices : caller phone rings, eventually confirms
  * the call, and is put through the callee
  * @param from : caller's number in international format.
  * @param to : callee's number in international format.
  * @param private (optional): withhold telephone number. Possible values are true,
  * false (by default), caller or callee.
  * @param max_duration (optional): maximum duration of a call Click-to-call, in seconds.
  * @param call_confirmation (optional): If set to true, allows the caller to confirm the call
  * before responding, by pressing the # key.
  * @param lang (optional): language used when call_confirmation is enabled.
  * @param audio_url (optional): the Url of an audio file played to caller.
  * @param notification_url (optional): the URL being notified in real time during the call.
  */
  $clickToCall->createCall( $from,
                            $to,
                            $private,
                            $max_duration,
                            $call_confirmation,
                            $lang,
                            $audio_url,
                            $notification_url
                          );
  //Response management
  if($clickToCall->getResponseCode()==200){
    echo" CallId : ".$clickToCall->getResponseCallId()."\n";
    echo"<span>clickToCall successfull</span>";
  }
  else {
    $exception = new Zend_Service_OrangeApi_Exception($clickToCall->getResponseCode()." -
".$clickToCall->getResponseMessage());
    $exception->DisplayFunctionalException();
  }
?>
```

UC-07

Create a conference call

Click-to-conf createConference sample

```
<?php
require_once('Zend/Service/OrangeApi/Exception.php');
require_once('Zend/Service/OrangeApi/ClickToConf.php');

 //mandatory data to create a Click-to-conf conference
  $accessKey = "my_access_key";
  $accesMode  = "access_mode";
 //mandatory data for caller mode
```

```
 $caller = "33123456789,33987654321";

 //Zend_Service_OrangeApi_ClickToConf initialization
 $clickToConf = new Zend_Service_OrangeApi_ClickToConf( $accessKey );

 /**
  * @desc create a conference call with incoming and outgoing calls
  * @param accesMode :
  *  room : access to the conference is controlled by a DTMF code
  *  caller : access to the conference is limited to a list of caller numbers
  *    room+caller : both
  *    none : access to the conference is only with outgoing calls via the createCall method
  * @param callers (mandatory for caller mode): list of numbers authorised to enter the conference.
  * Telephone numbers are in international format, e.g. 331XXXXXXXX separate by ",".
  * @param duration (optional): Conference duration (in minutes). Integer from 1 to 120.
  * Default value is 60.
  * @param when (optional): conference start date and time.
  * Date Format : YYYY-MM-DDTHH:MM:SSZ (see ISO 8601 format).
  * @param greeting (optional): language of conference (welcome greeting, announcements)
  */

  $clickToConf->createConference( $accesMode,
                                  $callers,
                                  $duration,
                                  $when,
                                  $greeting
                         );
 //Response management
 switch ($accesMode){

   case 'none' :
    //case none Mode
    if($clickToConf->getResponseCode()==200){
     echo " ConfId : ".$clickToConf->getResponse()->conference->id;
    }
   break;

   case 'room' :
     //case room Mode
     if($clickToConf->getResponseCode()==200){
       echo " ConfId : ".$clickToConf->getResponse()->conference->id;
       echo" Your Conference number is: ".
       $clickToConf->getResponse()->conference->room_restricted_access->conf_number."<br />".
       "Your Conference password is: ".
       $clickToConf->getResponse()->conference->room_restricted_access->room_number."<br />";
     }
   break;

   case 'caller' :
   //case caller Mode
     if($clickToConf->getResponseCode()==200){
       echo " ConfId : ".$clickToConf->getResponse()->conference->id;
foreach($clickToConf->getResponse()->conference->caller_restricted_access->caller as
               $caller){
         echo"Caller number : ".$caller->caller_number.
  "Conference number : ".$caller->conf_number.".<br />";
}
     }
   break;

   case 'room+caller' :
   //case room+caller Mode
   if($clickToConf->getResponseCode()==200){
    echo " ConfId : ".$clickToConf->getResponse()->conference->id;
     foreach($clickToConf->getResponse()->conference->caller_restricted_access->caller as $caller){
       echo"Caller number : ".$caller->caller_number;
       echo"Conference number :". $caller->conf_number;
     }
   }
```

```
    break;
  }
?>
```

UC-08

Add a caller to an existing conference call

Click-to-conf addCaller sample

```
<?php
require_once('Zend/Service/OrangeApi/Exception.php');
require_once('Zend/Service/OrangeApi/ClickToConf.php');

  //mandatory data to add callers to a Click-to-conf conference
   $accessKey = "myaccesskey";
   //conference_id_return_by_create_conference
   $conferenceId = "conf:a1234b56c78deee9f101";
   $addCallerNumbers="33123456789,33987654321";

  //Zend_Service_OrangeApi_ClickToConf initialization
  $clickToConf = new Zend_Service_OrangeApi_ClickToConf( $accessKey );

   /**
   * @desc add a possible caller to an existing conference call
   * @param conferenceId : conference ID (returned by the "createConference" method).
   * @param addCallerNumbers : list of telephone numbers authorised to join the conference.
   * (international format)
   */
   $clickToConf->addCaller( $conferenceId,
                            $addCallerNumbers
                          );
  //Response management
  if($clickToConf->getResponseCode()==200){
    echo" CallId : ".$clickToConf->getResponseCallId()."\n";
    echo"<span> successfull</span>";
  }
  else {
    $exception = new Zend_Service_OrangeApi_Exception($clickToConf->getResponseCode()." -
".$clickToConf->getResponseMessage());
    $exception->DisplayFunctionalException();
  }
?>
```

UC-09

Create an outgoing call from a conference to a callee

Click-to-conf createCall sample

```
<?php
require_once('Zend/Service/OrangeApi/Exception.php');
require_once('Zend/Service/OrangeApi/ClickToConf.php');

  //mandatory data to createCall to a Click-to-conf conference
  $accessKey = "myaccesskey";

  //conference_id_return_by_create_conference or addCaller
  $conferenceId = "conf:a1234b56c78deee9f101";
  $callers ="33123456789,33987654321";

  //Zend_Service_OrangeApi_ClickToConf initialization
  $clickToConf = new Zend_Service_OrangeApi_ClickToConf( $accessKey );

  /**
   * @desc createcall
   * @param callers : telephone number in international format.
   * @param conferenceId : conference ID.
   * @param notificationUrl (optional): the URL being notified in real time during the call.
   */
   $clickToConf->createCall( $callers,
                             $conferenceId,
                             $notificationUrl
                     );

  //Response management
  if($clickToConf->getResponseCode()==200){
    echo" CallId : ".$clickToConf->getResponseCallId()."\n";
    echo"<span>createCall successfull</span>";
  }
  else {
    $exception = new Zend_Service_OrangeApi_Exception($clickToConf->getResponseCode()." -
".$clickToConf->getResponseMessage());
    $exception->DisplayFunctionalException();
  }
?>
```

UC-10

Make interactive voice conferences

InteractiveVoice createConference sample

```
<?php
require_once('Zend/Service/OrangeApi/Exception.php');
require_once('Zend/Service/OrangeApi/InteractiveVoice.php');

  //mandatory data to createConference
  $accessKey = "my_access_key";

  //Optional data to createConference
  $duration = "conference_duration";
  $when = "conference_start_date";

  //Zend_Service_OrangeApi_InteractiveVoice initialization
  $InteractiveVoice =new Zend_Service_OrangeApi_InteractiveVoice( $accessKey );
   /**
    * @desc createConference
    * @param duration : Allows you to specify the duration of the conference (in minutes).
    * @param when : Defines the conference start date and time.
    * values: date in ISO 8601 format : YYYY-MM-DDTHH:MM:SSZ
    * now : for immediate conference start
    */
  $InteractiveVoice->createConference($duration ,
          $when
         );

  //Response management
  if($InteractiveVoice->getResponseCode()==200){
 echo" ConfId :".$InteractiveVoice->getResponseConfId()."\n";
 echo" StartDate :".$InteractiveVoice->getResponseStartDate()."\n";
 echo" EndDate :".$InteractiveVoice->getResponseEndDate()."\n";
 echo"<span> Create conference successfull.</span>";
  }
  else {
    $exception = new Zend_Service_OrangeApi_Exception($InteractiveVoice->getResponseCode()." -
".$InteractiveVoice->getResponseMessage());
    $exception->DisplayFunctionalException();
  }
?>
```

UC-11

Make interactive voice outgoing calls

InteractiveVoice Connect sample

```
<?php
require_once('Zend/Service/OrangeApi/Exception.php');
require_once('Zend/Service/OrangeApi/InteractiveVoice.php');

  //mandatory data to createConference
  $accessKey = "my_access_key";
  $part1 = "First_participant_identifier";
  $part2 = "Second_participant_identifier";

  //Optional data to connect
  $part1_presented_id = "opposite";
  $part2_presented_id = "private";
  $notification_url = "notification_Url";
  $max_duration = "max_duration";


  //Zend_Service_OrangeApi_InteractiveVoice initialization
  $InteractiveVoice =new Zend_Service_OrangeApi_InteractiveVoice( $accessKey );
   /**
    * @desc interactiveVoice
    * @param partX : Identifier of the first participant.
    * values:Identifier in the format of: "shortname" in the case in which
    * the call is initiated or if the participant is a VXML script or a conference.
    * Identifier in the format: "fullname" in the case of redirecting a participant of type
"telephone" to a new session.
    *
    * @param partX_presented_id : Defines the information which is presented on the telephone of the
first participant.
    *   -value: opposite: the telephone number of the second participant is displayed
    *          (if the second participant is a VXML script or a conference, a technical number will be
displayed).
    *   -private: no number displayed (the call is masked).
    *   -common: a technical number (which cannot be dialed) is displayed.
    *   -profile: it you have set up one, your dedicated number is displayed.
    * default value: opposite
    * @param notification_url : Allows a URL to receive, in real time, information related to the
session (in the case of incoming calls, this URL can
    *                           be defined on the API configuration page) to be defined.
    * @param max_duration : Specifies the maximum duration of a session, expressed in seconds.
    *         Once reached the call is disconnected.
    */
  $InteractiveVoice->connect($part1 ,
        $part2 ,
        $part1_presented_id ,
        $part2_presented_id ,
        $notification_url ,
        $max_duration
        );

  //Response management
  if($InteractiveVoice->getResponseCode()==200){
 echo" CallId : ".$InteractiveVoice->getResponseCallId()."\n";
 echo" participant 1 :".$InteractiveVoice->getResponsePart1()."\n";
 echo" participant 2 :".$InteractiveVoice->getResponsePart2()."\n";
 echo" participant 3 :".$InteractiveVoice->getResponsePart3()."\n";
 echo" participant 4 :".$InteractiveVoice->getResponsePart4()."\n";
 echo"<span> Create conference successfull.</span>";
  }
  else {
    $exception = new Zend_Service_OrangeApi_Exception($InteractiveVoice->getResponseCode()." -
".$InteractiveVoice->getResponseMessage());
    $exception->DisplayFunctionalException();
  }

?>
```

UC-12

Make interactive voice mute a participant

InteractiveVoice Mute sample

```
<?php
require_once('Zend/Service/OrangeApi/Exception.php');
require_once('Zend/Service/OrangeApi/InteractiveVoice.php');

  //mandatory data to createConference
  $accessKey = "my_access_key";
  $part_id = "participant_id_to_mute";
  $mute = true;

  //Zend_Service_OrangeApi_InteractiveVoice initialization
  $InteractiveVoice =new Zend_Service_OrangeApi_InteractiveVoice( $accessKey );

  /**
   *@desc The method muteParticipant disables the microphone of a participant (of type "telephone")
during a conférence.
   * Notes :
   *  - A participant on mute can still hear the other participants.
   *  - A participant on mute stays on mute even if he or she changes sessions.
   *@param part_id  : Unique identifier ("fullname") of the participant for whom the microphone is to
be disabled.
   *@param mute   : Allows enabling and disabling the microphone.
   * values:
   *  true: microphone disabled
   *  false: microphone enabled
   */

  $InteractiveVoice->mute($part_id ,
     $mute
    );

  //Response management
  if($InteractiveVoice->getResponseCode()==200){
 echo"<span> Mute successfull.</span>";
  }
  else {
    $exception = new Zend_Service_OrangeApi_Exception($InteractiveVoice->getResponseCode()." -
".$InteractiveVoice->getResponseMessage());
    $exception->DisplayFunctionalException();
  }
?>
```

UC-13

Make interactive voice release calls

InteractiveVoice ReleaseCall sample

```
<?php
require_once('Zend/Service/OrangeApi/Exception.php');
require_once('Zend/Service/OrangeApi/InteractiveVoice.php');

  //mandatory data to createConference
  $accessKey = "my_access_key";

  //Zend_Service_OrangeApi_InteractiveVoice initialization
  $InteractiveVoice =new Zend_Service_OrangeApi_InteractiveVoice( $accessKey );

  /**
   *@desc The method muteParticipant disables the microphone of a participant (of type "telephone")
during a conférence.
   * Notes :
   *  - A participant on mute can still hear the other participants.
   *  - A participant on mute stays on mute even if he or she changes sessions.
   *@param part_id  : Unique identifier ("fullname") of the participant for whom the microphone is to
be disabled.
   *@param mute   : Allows enabling and disabling the microphone.
   * values:
   *  true: microphone disabled
   *  false: microphone enabled
   */

  $InteractiveVoice->mute($part_id ,
     $mute
    );

  //Response management
  if($InteractiveVoice->getResponseCode()==200){
 echo"<span> Release successfull.</span>";
  }
  else {
    $exception = new Zend_Service_OrangeApi_Exception($InteractiveVoice->getResponseCode()." -
".$InteractiveVoice->getResponseMessage());
    $exception->DisplayFunctionalException();
  }
?>
```

UC-14

Make WebPhone calls

WebPhone sample

```
<?php
require_once('Zend/Service/OrangeApi/Exception.php');
require_once('Zend/Service/OrangeApi/WebPhone.php');

  //mandatory data to WebPhone
  $accessKey = "my_access_key";
  $to="callee_number";

  //Zend_Service_OrangeApi_WebPhone initialization
  $oWebPhoneClient = new Zend_Service_OrangeApi_WebPhone($accessKey);

 /**
   * @desc webPhone method
   * @param String $to <br />
   * Destination mobile number in international format.<br />
   * ex. : 3312345678 (France) or 441234567890 (UK)<br />
   * @param String $max_duration <br />
   * Maximum duration in seconds
   * @return ServiceResponse
   */
   $oWebPhoneClient->webPhone($to,
                              $max_duration
                             );

  //Response management
  if($oWebPhoneClient->getResponseCode()==200){
    echo' Call_id : '.$oWebPhoneClient->getResponseCallId()."\n".
 ' Status_message : '.$oWebPhoneClient->getResponseMessage()."\n".
 ' Status_code : '.$oWebPhoneClient->getResponseCode()."\n".
 ' Token : '.$oWebPhoneClient->token."\n".
 ' Gateway_url : '.$oWebPhoneClient->gateway_url."\n".
 ' Expiration_date : '.$oWebPhoneClient->expiration_date."\n";

  }
  else {
    $exception = new Zend_Service_OrangeApi_Exception($WebPhone->getResponseCode()." -
".$WebPhone->getResponseMessage());
    $exception->DisplayFunctionalException();
  }
?>
```

UC-15

Get details for a previous API request

getCallDetails sample

```
<?php
require_once('Zend/Service/OrangeApi/Exception.php');
require_once('Zend/Service/OrangeApi/CallDetails.php');

  //mandatory data to CallDetails
  $accessKey = "my_access_key";

  //Zend_Service_OrangeApi_CallDetails initialization
  $callDetails = new Zend_Service_OrangeApi_CallDetails( $accessKey );

  /**
   * @desc getCallDetails() : Calls the getCallDetails API method
   * @param callId (mandatory): Allows users to get call details on a specific identifier.
   * You can retrieve a callId in the orangeApi.log file,
   *  after sending a sms with ack parameter to true or check.
   */

  $callDetails->getCallDetails($callId);

  //Response management
  if($callDetails->getResponseCode()==200){
  echo"<span id='status'>Get Call Details successfull</span>";
  $log = new Zend_Service_OrangeApi_Common_Utility();
  $log->log(' CallId : '.$callDetails->getResponseCallId()."\n");
  $log->log(' Status : GetCallDetails successfull'."\n".$sep);
  $data=$callDetails->getXmlReturn()->data;

  //Display results
  echo"<div id='formatedResult'>";
  echo"<span style='margin-left: 50px;'><big><b>CallDetails</b></big></span><br />";
  foreach($data->api_call as $row){
   echo"<table class='result'>";
    if($row->call_id!=''){echo"<tr><td class='O'>Call id :</td><td
class='b'>$row->call_id</td></tr>";}
    if($row->date!=''){echo"<tr><td class='O'>Date :</td><td class='b'>$row->date</td></tr>";}
    if($row->from!=''){echo"<tr><td class='O'>From :</td><td class='b'>$row->from</td></tr>";}
    if($row->to!=''){echo"<tr><td class='O'>TO :</td><td class='b'>$row->to</td></tr>";}
    if($row->ip!=''){echo"<tr><td class='O'>IP :</td><td class='b'>$row->ip</td></tr>";}
    if($row->verb!=''){echo"<tr><td class='O'>Verb:</td><td class='b'>$row->verb</td></tr>";}
    if($row->cost!=''){echo"<tr><td class='O'>Cost:</td><td class='b'>$row->cost</td></tr>";}
    if($row->return_code!=''){echo"<tr><td class='O'>Return code :</td><td
class='b'>$row->return_code</td></tr>";}
    if($row->ack_status!=''){echo"<tr><td class='O'>Ack Status :</td><td
class='b'>$row->ack_status</td></tr>";}
    if($row->ack_status_msg!=''){echo"<tr><td class='O'>Ack Status message :</td><td
class='b'>$row->ack_status_msg</td></tr>";}
    if($row->ack_date!=''){echo"<tr><td class='O'>Ack Date:</td><td
class='b'>$row->ack_date</td></tr>";}
    if($row->ack_status_code!=''){echo"<tr><td class='O'>Ack status code:</td><td
class='b'>$row->ack_status_code</td></tr>";}
    if($row->tag!=''){echo"<tr><td class='O'>Tag:</td><td class='b'>$row->tag</td></tr>";}

    echo"</table><hr>";
  }
  echo"</div>";
 }
 else{
  echo"<span id='status'> Warning : ".$callDetails->getResponseCode()." -
".$callDetails->getResponseMessage()."</span>";
  $log = new Zend_Service_OrangeApi_Common_Utility();
  $log->log($callDetails->getResponseCode()." - ".$callDetails->getResponseMessage()."\n".$sep);
  header("Refresh: 10;url=index.php");
 }
?>
```

9. Class Skeletons

#1.Zend_Service_OrangeApi_ClientAbstract

```
require_once('Zend/Rest/Client.php');
require_once('Zend/Http/Client/Adapter/Proxy.php');
require_once( 'Zend/Service/OrangeApi/Exception.php' );

abstract class Zend_Service_OrangeApi_ClientAbstract extends Zend_Rest_Client
{
private $_path,$_id,$_apiParams,$_apiFiles,$_apiType,$_proxyAdapter,$_rawXmlReturn,$_xmlReturn;
private $_httpMethod;

public function __construct( $id, $httpMethod ) { ... }
public function getXmlReturn() { ... }
public function getRawXmlReturn() { ... }
public function setProxy( $proxyHost, $proxyPort, $proxyUser, $proxyPass ){ ... }
protected function setPath( $path ) { ... }
protected function setApiParams( $apiParams ) { ... }
protected function setApiFiles($apiFiles){ ... }
protected function setApiType($apiType){ ... }
public function clearApiParams() { ... }
protected function callApi() { ... }
private function removeCdata( $xmlStr ) { ... }
}
```

#2.Zend_Service_OrangeApi_Exception

```
require_once 'Zend/Service/Exception.php';
class Zend_Service_OrangeApi_Exception extends Zend_Service_Exception
{
  public function DisplayTechnicalException(){ ... }
  public function DisplayFunctionalException(){ ... }
}
```

#3.Zend_Service_OrangeApi_Sms

```
require_once('Zend/Service/OrangeApi/ClientAbstract.php');
require_once('Zend/Service/OrangeApi/Exception.php');

class Zend_Service_OrangeApi_Sms extends Zend_Service_OrangeApi_ClientAbstract
{
  private $_responseCode, $_responseMessage, $_responseCallId;
  public function __construct( $id ) { ... }
  public function sendSms( $from,
                           $to,
                           $content,
                           $ack
                         ) { ... }
  public function getResponseCallId() { ... }
  public function getResponseMessage() { ... }
  public function getResponseCode() { ... }
}
```

#4.Zend_Service_OrangeApi_Location

```
require_once( 'Zend/Service/OrangeApi/ClientAbstract.php' );
require_once( 'Zend/Service/OrangeApi/Exception.php' );

class Zend_Service_OrangeApi_Location extends Zend_Service_OrangeApi_ClientAbstract
{
private $_x, $_y, $_rad, $_responseCode, $_responseMessage, $_responseCallId;

public function __construct( $id ) { ... }
public function createAuthorization( $number ) { ... }
public function getLocation( $number ) { ... }
public function getX() { ... }
public function getY() { ... }
public function getRad() { ... }
public function getResponseCallId() { ... }
public function getResponseMessage() { ... }
public function getResponseCode() { ... }
}
```

#5.Zend_Service_OrangeApi_ClickToCall

```
require_once( 'Zend/Service/OrangeApi/ClientAbstract.php' );
require_once( 'Zend/Service/OrangeApi/Exception.php' );

class Zend_Service_OrangeApi_ClickToCall extends
Zend_Service_OrangeApi_ClientAbstract
{
  private $_responseCode,$_responseMessage,$_responseCallId;
  public function __construct( $id ) { ... }
  public function createCall( $from,
                              $to,
                              $private,
                              $max_duration,
                              $call_confirmation,
                              $lang,
                              $audio_url,
                              $notification_url
                            ){ ... }
  public function releaseCall ( $callId ){ ... }
  public function getResponseMessage() { ...}
  public function getResponseCode() { ... }
  public function getResponseCallId() { ... }
}
```

#6.Zend_Service_OrangeApi_ClickToConf

```
require_once( 'Zend/Service/OrangeApi/ClientAbstract.php' );
require_once( 'Zend/Service/OrangeApi/Exception.php' );

class Zend_Service_OrangeApi_ClickToConf extends
Zend_Service_OrangeApi_ClientAbstract
{
  private $_confId, $_roomModeConfNumber, $_roomNumber, $_callerModeConfNumber;
  private $_responseCode,$_responseMessage,$_response;

public function __construct( $id ) { ... }

public function createConference( $access_mode,
                                  $caller_numbers,
                                  $duration,
                                  $when,
                                  $greeting
                                ){ ... }

public function addCaller( $conf_id, $caller_numbers )  { ... }
public function createCall( $from, $to, $notification_url ) { ... }
public function getConfId() { ... }
public function getCallerModeConfNumber() { ... }
public function getRoomModeConfNumber() { ... }
public function getRoomNumber() { ... }
public function getResponseCallId() { ... }
public function getResponseMessage() { ... }
public function getResponseCode() { ... }
public function getResponse() { ... }
}
```

#7.Zend_Service_OrangeApi_CallDetails

```
require_once( 'Zend/Service/OrangeApi/ClientAbstract.php' );
require_once( 'Zend/Service/OrangeApi/Exception.php' );

class Zend_Service_OrangeApi_CallDetails extends
      Zend_Service_OrangeApi_ClientAbstract
{
  private $_responseCode,$_responseMessage,$_returnedXml,$_responseCallId;
  public function __construct( $id ) { ... }
  public function getCallDetails($callId) { ... }
  public function getResponseCallId() { ... }
  public function getResponseMessage() { ... }
  public function getResponseCode() { ... }
  public function getXmlResponse() { ... }
}
```

#8.Zend_Service_OrangeApi_MMS

```
require_once('Zend/Service/OrangeApi/ClientAbstract.php');
require_once('Zend/Service/OrangeApi/Exception.php');

class Zend_Service_OrangeApi_Mms extends Zend_Service_OrangeApi_ClientAbstract
{
  private $_responseCode,$_responseMessage ,$_responseCallId;
  public function __construct( $id ) { ... }
  public function sendMms( $from,
                           $to,
                           $content,
                           $subject,
                           $image,
                           $sound,
                           $video ) { ... }
  public function getResponseCallId() { ... }
  public function getResponseMessage() { ... }
  public function getResponseCode() { ... }

}
```

#9.Zend_Service_OrangeApi_InteractiveVoice

```
require_once( 'Zend/Service/OrangeApi/ClientAbstract.php' );
require_once( 'Zend/Service/OrangeApi/Exception.php' );

class Zend_Service_OrangeApi_InteractiveVoice extends
      Zend_Service_OrangeApi_ClientAbstract
{
  private $_responseCode,$_responseMessage,$_returnedXml,
   $_responseConfId,$_responseCallId,$_responseEndDate,
   $_responseStartDate,$_responsePart1,$_responsePart2,
   $_responsePart3,$_responsePart4;

  public function __construct( $id ) { ... }
  public function connect($part1,
     $part2,
     $part1_presented_id,
     $part2_presented_id,
     $notification_url,
     $max_duration) { ... }
  public function createConference($duration,$when){ ... }
  public function mute($part_id,$mute){ ... }
  public function releaseCall($callId){ ... }

  public function getResponseMessage() { ... }
  public function getResponseCode() { ... }
  public function getResponseCallId() { ... }
  public function getResponseConfId() { ... }
  public function getResponsePart1() { ... }
  public function getResponsePart2() { ... }
  public function getResponsePart3() { ... }
  public function getResponsePart4() { ... }
  public function getResponseStartDate(){ ... }
  public function getResponseEndDate(){ ... }
}
```

#10.Zend_Service_OrangeApi_WebPhone

```
require_once( 'Zend/Service/OrangeApi/ClientAbstract.php' );
require_once( 'Zend/Service/OrangeApi/Exception.php' );

class Zend_Service_OrangeApi_WebPhone extends
      Zend_Service_OrangeApi_ClientAbstract
{
  private $_responseCode,$_responseMessage,$_returnedXml;
  public $token,$gateway_url,$expiration_date;
  public function __construct( $id ) { ... }
  public function webPhone($to,
                           $max_duration
                          ) { ... }
  public function getResponseCallId() { ... }
  public function getResponseMessage() { ... }
  public function getResponseCode() { ... }
}
```

]]></ac:plain-text-body></ac:macro>
]]></ac:plain-text-body></ac:macro>