Tutorial de

Integração

PHP

Sumário

[Tutorial de Integração - PHP 3](#_Toc394593458)

[Configurando Envio 4](#_Toc394593459)

[Envio de SMS individual 5](#_Toc394593460)

[Consulta de status de SMS individual 7](#_Toc394593461)

[Envio de SMS múltiplo 8](#_Toc394593462)

[Cancelamento de SMS agendado 10](#_Toc394593463)

[Consulta de SMS recebidos 11](#_Toc394593464)

[Consulta avançada de SMS recebidos 12](#_Toc394593465)

[Glossário 13](#_Toc394593466)

Tutorial de Integração - PHP

Objetivo

O objetivo do tutorial de integração é guiar os desenvolvedores que desejam integrar suas aplicações ao gateway de SMS da Zenvia, automatizando assim seus envios de SMS conforme sua necessidade.

Mostraremos em simples passos como fazer suas aplicações PHP enviarem SMS por meio da API de integração REST que disponibilizamos para facilitar os envios.

Pré-requisitos

Para executar a biblioteca de integração, basta ter instalado um servidor HTTP que interprete PHP, tal como Apache, TomCat, entre outros.

Possuir a extensão [JSON](http://php.net/manual/pt_BR/book.json.php) instalada e habilitada. Para versões do PHP abaixo da 5.2 a extensão pode ser instalada via pecl. Para versões do PHP acima da 5.2 a extensão JSON já vem habilitada por default.

Possuir o módulo [OpenSSL](http://php.net/manual/pt_BR/book.openssl.php) instalado e habilitado.

Entendendo a biblioteca PHP

A biblioteca PHP é um conjunto de classes que tem como principal função fazer requisições REST da forma mais organizada e simplificada possível, fazendo com que poucas informações sejam passadas por sua aplicação.

Configurando Envio

1º Passo

Faça o *download* da biblioteca de integração REST por meio do *link* <http://www.zenvia.com.br/desenvolvedores/bibliotecas/REST/PHP.zip>

2º Passo

Para manipulação de SMS (envios e consultas), inclua em seu código PHP o arquivo autoload.php, que se encontra na raiz do diretório da biblioteca de integração.

Exemplo:

|  |
| --- |
| **include\_once**("php-rest-api/autoload.php"); |

3º Passo

Para qualquer operação na API de integração, é preciso identificar-se com sua conta e código de acesso.

Trabalharemos com a instância da classe SmsFacade para manipulação de SMS (envio e consulta), passando por parâmetro de seu construtor a sua conta e o código de acesso. Tais dados são cedidos pela área de produtos da Zenvia juntamente com a negociação do contrato firmado com a empresa.

Exemplo:

|  |
| --- |
| $smsFacade = **new** SmsFacade("alias.fornecido", "senha.fornecida"); |

Envio de SMS individual

1º Passo

Instancie a classe *SmsFacade*,informando a sua conta e senha.

Exemplo:

|  |
| --- |
| $smsFacade = new SmsFacade("conta", "senha"); |

2º Passo

Instancie a classe *Sms* e adicione as informações: [*to*](#id.377oi69ljv0) (obrigatório), *msg* (obrigatório), [*from*](#id.uo3hfjajbztm) (opcional), [*id*](#id.d9bbw4aadn8w) (opcional), [*schedule*](#id.hnfvd7ls7ra6) (opcional) e [*callback*](#id.ou7x7hr5l1gi) (opcional).

Exemplo:

|  |
| --- |
| $sms = **new** Sms();  $sms->setTo("555199990101");  $sms->setMsg("Minha mensagem de teste");  //Deve estar no formato definido na ISO 8601(Y-m-dTH:i:s)  $sms->setSchedule("2014-07-13T16:00:00"); |

3º Passo

Chame o método *send* passando por parâmetro o objeto *$sms criado anteriormente*.

Exemplo:

|  |
| --- |
| try{  $response = $smsFacade->send($sms);  echo "Status: ".$response->getStatusCode() . " - " . $response->getStatusDescription();  echo "\nDetalhe: ".$response->getDetailCode() . " - " . $response->getDetailDescription();  }  catch(Exception $ex ){  echo $ex->getMessage();  } |

**Obs.**: Caso a sua conta possua a configuração de agrupadores habilitada será necessário informar o id do agrupador no momento do envio.

Exemplo:

|  |
| --- |
| try{  $aggregateId=1456977;  $response = $smsFacade->send($sms, $aggregateId);  }  catch(Exception $ex ){  echo $ex->getMessage();  } |

4º Passo

Obter e tratar a resposta da requisição. O método *send* retorna um objeto do tipo SmsResponse com o status e o detalhe da operação.

Exemplo:

|  |
| --- |
| echo "Status: ".$response->getStatusCode() . " - " . $response->getStatusDescription();  echo "\nDetalhe: ".$response->getDetailCode() . " - " . $response->getDetailDescription(); |

Consulta de status de SMS individual

1º Passo

Instancie a classe *SmsFacade* informando a sua conta e senha.

Exemplo:

|  |
| --- |
| $smsFacade = new SmsFacade("conta", "senha"); |

2º Passo

Chame o método *getStatus* passando por parâmetro o [*id*](#id.d9bbw4aadn8w) do SMS que deseja consultar.

Exemplo:

|  |
| --- |
| $id = "12cbe14f";  try{  $response = $smsFacade->getStatus($id);  }  catch(Exception $ex){  echo $ex->getMessage();  } |

3º Passo

Obter e tratar a resposta da requisição. O método *getStatus* retorna um objeto do tipo SmsStatusResponse com o status e o detalhe da operação. Caso a mensagem com o id passado seja encontrada, outras informações como a data de recebimento do SMS são retornadas.

Exemplo:

|  |
| --- |
| //Código e descrição do status atual da mensagem  echo "Status: " . $response->getStatusCode() . " - " . $response->getStatusDescription();  //Código e descrição do detalhe do status atual da mensagem  echo "\nDetalhe: " . $response->getDetailCode() . " - " . $response->getDetailDescription();  if ($response->getStatusCode() == "00") {  //Id da mensagem  echo "\nId: " . $response->getId();  //Data de recebimento da mensagem no celular  echo "\nRecebido em: " . $response->getReceived();  } |

Envio de SMS múltiplo

1º Passo

Instancie a classe *SmsFacade*,informando a sua conta e senha.

Exemplo:

|  |
| --- |
| $smsFacade = new SmsFacade("conta", "senha"); |

2º Passo

Criar um objeto do tipo Sms para cada mensagem que deseja enviar e adicioná-los a um array.

Exemplo:

|  |
| --- |
| $sms1 = new Sms();  $sms1->setTo("555199999991");  $sms1->setMsg("Esta é a primeira mensagem de exemplo do envio múltiplo utilizando a api PHP.");  $sms2 = new Sms();  $sms2->setTo("555199999992");  $sms2->setMsg("Esta é a segunda mensagem de exemplo do envio múltiplo utilizando a api PHP.");  $smsList = array($sms1, $sms2); |

3º Passo

Chame o método *sendMultiple* passando por parâmetro o array *$smsList* com a lista de mensagens que deseja enviar.

Exemplo:

|  |
| --- |
| try{  $responses = $smsFacade->sendMultiple($smsList);  }  catch(Exception $ex ){  echo $ex->getMessage();  } |

**Obs.**: Caso a sua conta possua a configuração de agrupadores habilitada será necessário informar o id do agrupador no momento do envio.

Exemplo:

|  |
| --- |
| try{  $aggregateId=123456;  $responses = $smsFacade->sendMultiple($smsList, $aggregateId);  }  catch(Exception $ex ){  echo $ex->getMessage();  } |

4º Passo

Obter e tratar a resposta da requisição. O método *sendMultiple* retorna um array de objetos do tipo SmsResponse.

Exemplo:

|  |
| --- |
| foreach ($responses as $response) {  echo "Status: " . $response->getStatusCode() . " - " . $response->getStatusDescription();  echo "\nDetalhe: " . $response->getDetailCode() . " - " . $response->getDetailDescription() . "\n";  } |

Cancelamento de SMS agendado

1º Passo

Instancie a classe *SmsFacade* informando a sua conta e senha.

Exemplo:

|  |
| --- |
| $smsFacade = new SmsFacade("conta", "senha"); |

2º Passo

Chame o método *cancel*, passando por parâmetro o [*id*](#id.d9bbw4aadn8w) do SMS que deseja cancelar.

Exemplo:

|  |
| --- |
| $id = "12cbe14f";  try{  $response = $smsFacade->cancel($id);  }  catch(Exception $ex){  echo $ex->getMessage();  } |

3º Passo

Obter e tratar a resposta da requisição. O método *cancel* retorna um objeto do tipo SmsResponse com o status e o detalhe da operação.

Exemplo:

|  |
| --- |
| echo "Status: ".$response->getStatusCode() . " - " . $response->getStatusDescription();  echo "\nDetalhe: ".$response->getDetailCode() . " - " . $response->getDetailDescription(); |

Consulta de SMS recebidos

1º Passo

Instancie a classe *SmsFacade* informando a sua conta e senha.

Exemplo:

|  |
| --- |
| $smsFacade = new SmsFacade("conta", "senha"); |

2º Passo

Chame o método *listMessagesReceived*.

Exemplo:

|  |
| --- |
| try{  //resposta contendo o status da operação e a listagem de mensagens recebidas.  $response = $smsFacade->listMessagesReceived();  }  catch(Exception $ex){  echo $ex->getMessage();  } |

3º Passo

Obter e tratar a resposta da requisição. O método *listMessagesReceived* retorna um objeto do tipo SmsReceivedResponse com o qual é possível recuperar o status da operação e as mensagens recebidas.

Exemplo:

|  |
| --- |
| if ($response->hasMessages()) {  $messages = $response->getReceivedMessages();  foreach ($messages as $smsReceived) {  echo "\nCelular: " . $smsReceived->getMobile();  echo "\nData de recebimento: " . $smsReceived->getDateReceived();  echo "\nMensagem: " . $smsReceived->getBody();  //Id da mensagem que originou a mensagem de resposta  echo "\nId da mensagem de origem: " . $smsReceived->getSmsOriginId();  }  } |

Consulta avançada de SMS recebidos

1º Passo

Instancie a classe *SmsFacade* informando a sua conta e senha.

Exemplo:

|  |
| --- |
| $smsFacade = new SmsFacade("conta", "senha"); |

2º Passo

Chame o método *searchMessagesReceived passando como parâmetro:* o período inicial(obrigatório), o período final(obrigatório), o celular(opcional) e o id do SMS(opcional).

Exemplo:

|  |
| --- |
| try{  //Período inicial no formato ISO 8601(Y-m-dTH:i:s)  $startPeriod="2014-07-13T00:00:00";  //Período final no formato ISO 8601(Y-m-d\TH:i:s)  $endPeriod="2014-07-13T23:59:59";  //Celular que enviou a mensagem para plataforma.  $mobile="555199999991";  //Id da mensagem SMS que originou a mensagem de resposta.  $smsId="abc123fgff";  //resposta contendo o status da operação e a listagem de mensagens recebidas.  $response = $smsFacade->searchMessagesReceived($startPeriod, $endPeriod, $mobile, $smsId);  }  catch(Exception $ex){  echo $ex->getMessage();  } |

3º Passo

Obter e tratar a resposta da requisição. O método *searchMessagesReceived* retorna um objeto do tipo SmsReceivedResponse com o qual é possível recuperar o status da operação e as mensagens recebidas.

Exemplo:

|  |
| --- |
| if ($response->hasMessages()) {  $messages = $response->getReceivedMessages();  foreach ($messages as $smsReceived) {  echo "\nCelular: ". $smsReceived->getMobile();  echo "\nData de recebimento: " . $smsReceived->getDateReceived();  echo "\nMensagem: " . $smsReceived->getBody();  //Id da mensagem que originou a mensagem de resposta  echo "\nId da mensagem de origem: " . $smsReceived->getSmsOriginId();  }  } |

Glossário

|  |  |  |
| --- | --- | --- |
| **Item** | **Propriedade** | **Descrição** |
| 1 | *To* | Número de telefone do SMS no formato DDI + DDD + Telefone (Exemplo: 555199990101). |
| 2 | *Msg* | Mensagem que será enviada ao telefone. Terá no máximo 160 caracteres (sem o campo From). |
| 3 | *From* | Identificação do remetente que será atribuído à mensagem. O campo "Message" + o campo "From" devem ter tamanho máximo de 150 caracteres. |
| 4 | *Id* | Sua identificação do SMS para fins de consulta. |
| 5 | *Schedule* | Data de agendamento de envio do SMS. Deve estar no formato descrito na ISO 8601 |
| 6 | Tipos de *callback* | Tipo de callback que a sua aplicação terá como retorno. São definidos na classe Sms como as seguintes constantes:   * **CALLBACK\_NONE** ("NONE") (Padrão): Não será enviada mensagem de *callback* para a sua aplicação. * **CALLBACK\_FINAL** ("FINAL"): Será enviada a mensagem de *callback* com somente o estado final de cada mensagem enviada. * **CALLBACK\_ALL** ("ALL"): Será enviada a mensagem de *callback* com os estados intermediários e o estado final de cada mensagem enviada.   **Observação:** Para os *callbacks* FINAL e ALL, deverá ser cadastrado no Atendimento da Zenvia um *host* ao qual serão enviadas as requisições. |
| 7 | *AggregateId* | Id do agrupador para as contas que possuírem a configuração de agrupadores habilitada. |