Zend_Service_Chargify - Dan Bowen

<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_Chargify Component Proposal

Proposed Component Name	Zend_Service_Chargify
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Service_Chargify
Proposers	Dan Bowen
Zend Liaison	TBD
Revision	1.0 - 19 May 2011: Initial Draft. (wiki revision: 4)

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
9. Class Skeletons

1. Overview

Chargify.com is a subscription billing system that allows web application developers to charge customer credit cards on a recurring basis.

The Zend_Service_Chargify component makes it easy for developers to interact with the Chargify.com REST API for seamless integration into their website or application (as opposed to using hosted payment pages).

2. References

- Chargify API Documentation

3. Component Requirements, Constraints, and Acceptance Criteria

The component **will** wrap the entire Chargify REST API.
The component **will** provide response objects that implment ArrayAccess, Iterator, Countable for easy iteration.
The component **will** allow for both JSON and XML requests and responses.
The component **will** respond appropriately to HTTP response codes returned by the Chargify API
The component **will** provide an expressive API (setters) so it is obvious what parameters need to be sent with a given request.

4. Dependencies on Other Framework Components

- Zend_Service_Abstract
- Zend_Http_Client
- Zend_Http_Response
- Zend_Json

5. Theory of Operation

The Application Developer has an account with Chargify.com
The Application Developer instantiates Zend_Service_Chargify with their API key & password (and possibly other options)
The Application Developer uses API functionality to create customers, create subscriptions and view transaction history (and more)

6. Milestones / Tasks

Milestone 1: [DONE] Supporting Documentation, Use Cases, Class Skeletons, uploaded.
Milestone 2: Check in working Code to support the basic Zend_Service_Chargify instantiation
Milestone 3: Check in working code to support Products functionality
Milestone 4: Check in working code to support Components functionality
Milestone 5: Check in working code to support Coupons functionality
Milestone 6: Check in working code to support Customers functionality
Milestone 7: Check in working code to support Subscriptions functionality
Milestone 8: Check in working code to support Transactions functionality
Milestone 9: Check in working code to support Migrations functionality
Milestone 10: Check in working code to support Charges functionality
Milestone 11: Check in working code to support Adjustments functionality
Milestone 12: Check in working code to support Metered Usage functionality
Milestone 13: Check in working code to support Quantity Component Allocations functionality
Milestone 14: Check in working code to support Refunds functionality
Milestone 15: Check in working code to support Statements functionality
Milestone 16: Redesign and update code based off community feedback.
Milestone 17: PHPUnit Tests exist.
Milestone 18: Solidify DocBlox API documentation

7. Class Index

- Zend_Service_Chargify
- Zend_Service_Chargify_Abstract
- Zend_Service_Chargify_Adjustment
- Zend_Service_Chargify_Charge
- Zend_Service_Chargify_Component
- Zend_Service_Chargify_Coupon
- Zend_Service_Chargify_Customer
- Zend_Service_Chargify_Exception
- Zend_Service_Chargify_Product
- Zend_Service_Chargify_Refund
- Zend_Service_Chargify_Statement
- Zend_Service_Chargify_Subscription
- Zend_Service_Chargify_Transaction

8. Use Cases

UC-01

Create new customer and subscription at the same time

```
 <?php
// this could also be a Zend_Config object
$config = array(
  'format'   => 'json',
  'api_key'  => 'yourapikey',
  'hostname' => 'yoursite.chargify.com',
  'password' => 'x' // password is always x. might not need to send this as a config option
);
$service = new Zend_Service_Chargify($config);

// create customer and subscription
$sub = $service->subscription()
               ->setProductHandle('your-product-handle')
               ->setCustomerAttributes(array(
                 'first_name'   => 'First Name',
                 'last_name'    => 'Last Name',
                 'email'        => 'user@email.com',
                 'reference'    => $uniqueUserIdFromYourApp
               ))
               ->create();

// check for errors
if ($sub->isError()) {
  $errors = $sub->getErrors();
}
```

UC-02

Cancel a subscription

```
<?php
$service->subscription()->cancel(123456);

UC-03 List transactions for a specific subscription. Charges, refunds and payments only

$transactions = $service->transaction()
                        ->setPagination(1, 20)
                        ->setKinds(array('charge', 'refund', 'payment'))
                        ->listBySubscription(123456);
```

9. Class Skeletons

```
<?php
/**
 * @category   Zend
 * @package    Zend_Service_Chargify
```

```
 */
class Zend_Service_Chargify extends Zend_Service_Abstract
{
  /**
   * The complete hostname; e.g. "my-app-subdomain.chargify.com",
   * not just "my-app-subdomain"
   *
   * @var string
   */
  protected $_hostname;

  /**
   * Your api key
   *
   * @var string
   */
  protected $_apiKey;

  /**
   * Your http authentication password. The password is always "x" but for now we're
   * accepting in in the $config
   *
   * @var string
   */
  protected $_password;

  /**
   * xml or json
   *
   * @var string
   */
  protected $_format;

  /**
   * Original config used in constructor
   *
   * @var mixed Zend_Config|array
   */
  protected $_config;

  /**
   * Enter description here...
   *
   * @param mixed Zend_Config|array $config
   */
  public function __construct($config) {}

  /**
   * xml or json
   *
   * @return string
   */
  public function getFormat() {}

  /**
   * Returns original config sent in constructor
   *
   * @return Zend_Config|array
   */
  public function getConfig() {}

  /**
   * Sends a request to the Chargify API
   *
   * @param string $path
   * @param string $method; GET, POST, PUST, DELETE
   * @param string $rawData
   * @param array $params
   * @return Zend_Http_Response
   */
```

```
    public function request($path, $method, $rawData = NULL, $params = NULL) {}

    /**
     * Helper for instantiating a Zend_Service_Chargify_Customer object
     *
     * @return Zend_Service_Chargify_Customer
     */
    public function customer() {}

    /**
     * Helper for instantiating a Zend_Service_Chargify_Subscription object
     *
     * @return Zend_Service_Chargify_Subscription
     */
    public function subscription() {}

    /**
     * Helper for instantiating a Zend_Service_Chargify_Product object
     *
     * @return Zend_Service_Chargify_Product
     */
    public function product() {}

    /**
     * Helper for instantiating a Zend_Service_Chargify_Adjustment object
     *
     * @return Zend_Service_Chargify_Adjustment
     */
    public function adjustment() {}

    /**
     * Helper for instantiating a Zend_Service_Chargify_Charge object
     *
     * @return Zend_Service_Chargify_Charge
     */
    public function charge() {}

    /**
     * Helper for instantiating a Zend_Service_Chargify_Component object
     *
     * @return Zend_Service_Chargify_Component
     */
    public function component() {}

    /**
     * Helper for instantiating a Zend_Service_Chargify_Coupon object
     *
     * @return Zend_Service_Chargify_Coupon
     */
    public function coupon() {}

    /**
     * Helper for instantiating a Zend_Service_Chargify_Transaction object
     *
     * @return Zend_Service_Chargify_Transaction
     */
    public function transaction() {}

    /**
     * Helper for instantiating a Zend_Service_Chargify_Refund object
     *
     * @return Zend_Service_Chargify_Refund
     */
    public function refund() {}

    /**
     * Helper for instantiating a Zend_Service_Chargify_Statement object
     *
     * @return Zend_Service_Chargify_Statement
```

```
    */
  public function statement() {}
}
```

```
<?php
/**
 * @category   Zend
 * @package    Zend_Service_Chargify
 */
abstract class Zend_Service_Chargify_Abstract implements ArrayAccess, Iterator, Countable
{
  protected $_params = array();

  protected $_errors = array();

  /**
   * Data container for providing ArrayAccess on this object
   *
   * @var array
   */
  protected $_data = array();

  /**
   * Instance of Zend_Service_Chargify sent in constructor
   *
   * @var Zend_Service_Chargify
   */
  protected $_service;

  public function __construct(Zend_Service_Chargify $service) {}

  /**
   * Get Zend_Service_Chargify instance sent in constructor
   *
   * @return Zend_Service_Chargify
   */
  public function getService() {}

  /**
   * Set a single parameter. Provides fluent interface.
   *
   * @param string $param
   * @param mixed $value
   * @return Zend_Service_Chargify_Abstract
   */
  public function setParam($param, $value) {}

  /**
   * Get a single parameter.
   *
   * @param string $paramName
   * @return string|array
   */
  public function getParam($paramName) {}

  /**
   * Get all params.
   *
   * @return array
   */
  public function getParams() {}

  /**
   * Assembles xml from given array
   *
```

```
 * @param array $array
 * @return string
 */
public function arrayToXml($array) {}

/**
 * Assmbles a full xml document from given array
 *
 * @param array $array
 * @return string
 */
public function getXml($array) {}

/**
 * Assembles the raw data (xml or json) from the given array
 *
 * @param array $array
 * @return string
 */
public function getRawData($array) {}

/**
 * Helper to determine if there are errors with the request
 *
 * @return bool
 */
public function isError() {}

/**
 * Array of errors, if any, returned from Chargify.
 * @return array
 */
public function getErrors() {}

/**
 * Transfoms the response body (xml or json) into an array we can more easily
 * work with.
 *
 * @param Zend_Http_Response $response
 * @return array
 */
public function getResponseArray(Zend_Http_Response $response) {}

/**
 * Implementation of ArrayAccess
 */

/**
 * @param mixed $offset
 * @param mixed $value
```

```
    */
  public function offsetSet($offset, $value) {}
}
```

```
<?php
/**
 * @category   Zend
 * @package    Zend_Service_Chargify
 * @link http://docs.chargify.com/api-subscriptions
 * @link http://docs.chargify.com/api-migrations
 */
class Zend_Service_Chargify_Subscription extends Zend_Service_Chargify_Abstract
{
  /**
   * The API Handle of the product for which you are creating a subscription
   * or migrating to. Required, unless a product_id is given instead.
   *
   * @param string $handle
   * @return Zend_Service_Chargify_Subscription
   */
  public function setProductHandle($handle) {}

  /**
   * The ID of the Product that the subscription will created for or migrated
   * to.
   *
   * @param string|int $id
   * @return Zend_Service_Chargify_Subscription
   */
  public function setProductId($id) {}

  /**
   * The ID of an existing customer within Chargify. Required, unless a
   * customer_reference or a set of customer_attributes is given.
   *
   * @param string|int $idFromChargify
   * @return Zend_Service_Chargify_Subscription
   */
  public function setCustomerId($idFromChargify) {}

  /**
   * The reference value (provided by your app) of an existing customer within
   * Chargify. Required, unless a customer_id or a set of customer_attributes
   * is given.
   *
   * @param string|int $idFromYourApp
   * @return Zend_Service_Chargify_Subscription
   */
  public function setCustomerReference($idFromYourApp) {}

  /**
   * Array containing component IDs and quantities for components to be added
   * at subscription creation time. Array should look like this:
   *
   * array(
   *  array('component_id' => $id, 'allocated_quantity' => $quantity),
   *  array('component_id' => $id, 'allocated_quantity' => $quantity)
   * )
   *
   * @param array $components
   * @return Zend_Service_Chargify_Subscription
   * @todo Unit test this
   */
  public function setComponents($components) {}
```

```
/**
 * Set the attributes for the customer. Useful when creating customer and
 * subscription at the same time.
 *
 * @param array $attributes
 * @return Zend_Service_Chargify_Subscription
 */
public function setCustomerAttributes($attributes) {}

/**
 * Set payment profile attributes. Useful when accepting (or requiring) a
 * credit card at signup.
 *
 * @param array $attributes
 * @return Zend_Service_Chargify_Subscription
 */
public function setPaymentProfileAttributes($attributes) {}

/**
 * (Optional) Can be used when canceling a subscription (via the HTTP DELETE
 * method) to make a note about the reason for cancellation.
 *
 * @param string $message
 * @return Zend_Service_Chargify_Subscription
 */
public function setCancellationMessage($message) {}

/**
 * Required when using cancelDelayed(). Set the subscription to be cancelled
 * at the end of the current billing period.
 *
 * @param bool $bool
 * @return Zend_Service_Chargify_Subscription
 */
public function setCancelAtEndOfPeriod($bool) {}

/**
 * Only used for import.
 *
 * @param string $nextBilling
 * @return Zend_Service_Chargify_Subscription
 */
public function setNextbillingAt($nextBilling) {}

/**
 * Boolean, default 0. If 1 is sent initial charges will be assessed. If 0 is
 * sent initial charges will be ignored.
 *
 * @param int $initialCharge
 * @return Zend_Service_Chargify_Subscription
 */
public function setIncludeInitialCharge($initialCharge = 0) {}

/**
 * Boolean, default 0. If 1 is sent the customer will migrate to the new
 * product with a trial if one is available. If 0 is sent, the trial period
 * will be ignored.
 *
 * @param int $includeTrial
 * @return Zend_Service_Chargify_Subscription
 */
public function setIncludeTrial($includeTrial = 0) {}

/**
 * Set the coupon code to be used at subscription creation.
 *
 * @param string $coupon_code
 * @return Zend_Service_Chargify_Subscription
 * @link http://docs.chargify.com/api-coupons
 * @todo Unit test this
```

```
     */
    public function setCouponCode($coupon_code) {}

    /**
     * An integer value which specifies which page of results to fetch, starting
     * at 1. Fetching successively higher page numbers will return additional
     * results, until there are no more results to return (in which case an empty
     * result set will be returned). Defaults to 1.
     *
     * @param int $page
     * @return Zend_Service_Chargify_Subscription
     */
    public function setPage($page) {}

    /**
     * How many records to fetch in each request, defaults to 2000. Note:
     * fetching subscriptions is currently unoptimized, so fetching large batches
     * of subscriptions will be a slow operation.
     *
     * @param int $perPage
     * @return Zend_Service_Chargify_Subscription
     */
    public function setPerPage($perPage) {}

    /**
     * Get subscription data for given subscription ID
     *
     * @param int $subscriptionId
     * @return Zend_Service_Chargify_Subscription
     */
    public function read($subscriptionId) {}

    /**
     * Create a new subscription.
     *
     * @return Zend_Service_Chargify_Subscription
     * @see Zend_Service_Chargify_Subscription::setProductHandle()
     * @see Zend_Service_Chargify_Subscription::setProductId()
     * @see Zend_Service_Chargify_Subscription::setCustomerId()
     * @see Zend_Service_Chargify_Subscription::setCustomerAttributes()
     * @see Zend_Service_Chargify_Subscription::setPaymentProfileAttributes()
     * @see Zend_Service_Chargify_Subscription::setCouponCode()
     * @see Zend_Service_Chargify_Subscription::setCustomerReference()
     * @see Zend_Service_Chargify_Subscription::setNextBillingAt()
     * @link http://docs.chargify_Subscription.com/api-subscriptions
     * @link http://docs.chargify.com/api-coupons
     */
    public function create() {}

    /**
     * Set the given subscription ID to be canceled immediately
     *
     * @param int $id
     * @return Zend_Service_Chargify_Subscription
     * @see @see Zend_Service_Chargify_Subscription::setCancellationMessage()
     */
    public function cancelImmediately($id) {}

    /**
     * Set the subscription to be canceled at the end of the current billing
     * period.
     *
     * @param int $id
     * @return Zend_Service_Chargify_Subscription
     * @see Zend_Service_Chargify_Subscription::setCancelAtEndOfPeriod()
     * @see Zend_Service_Chargify_Subscription::setCancellationMessage()
     */
    public function cancelDelayed($id) {}

    /**
```

```
 * Reactivate the given subscription ID
 *
 * @param int $id
 * @return Zend_Service_Chargify_Subscription
 * @see Zend_Service_Chargify_Subscription::setIncludeTrial()
 */
public function reactivate($id) {}

/**
 * Reset the subscription balance to $0.00
 *
 * @param int $subscriptionId
 * @return Zend_Service_Chargify_Subscription
 */
public function resetBalance($subscriptionId) {}

/**
 * Update the given subscription ID.
 *
 * For moving to a different product (WITHOUT PRORATION), changing customer
 * attributes or changing payment profile attributes.
 *
 * @param int $id
 * @return Zend_Service_Chargify_Subscription
 * @see Zend_Service_Chargify_Subscription::setProductHandle()
 * @see Zend_Service_Chargify_Subscription::setCustomerAttributes()
 * @see Zend_Service_Chargify_Subscription::setPaymentProfileAttributes()
 */
public function update($id) {}

/**
 * Migrate a subscription to a new product with prorated credits/debits.
 *
 * @param int $id
 * @return Zend_Service_Chargify_Subscription
 * @see Zend_Service_Chargify_Subscription::setProductId()
 * @see Zend_Service_Chargify_Subscription::setProductHandle()
 * @see Zend_Service_Chargify_Subscription::setIncludeTrial()
 * @see Zend_Service_Chargify_Subscription::setIncludeInitialCharge()
 * @link http://docs.chargify.com/api-migrations
 * @todo ?? should this be moved to Zend_Service_Chargify_Migration
 */
public function migrate($id) {}

/**
 * List subscriptions for a site.
 *
 * @return Zend_Service_Chargify_Subscription
 * @see Zend_Service_Chargify_Subscription::setPage()
 * @see Zend_Service_Chargify_Subscription::setPerPage()
 */
public function listSubscriptions() {}

/**
 * List subscriptions for a specific customer.
 *
 * @param int $customerId
 * @return Zend_Service_Chargify_Subscription
 * @see Zend_Service_Chargify_Subscription::setCustomerId()
 */
public function listByCustomer($customerId) {}

/**
 * When returning multiple products the array is different depending on which
 * format (xml/json) you are using. This normalizes the array for us so we can
 * rely on a consistent structure.
 *
 * @param array $responseArray
 * @return array
```

```
    */
  protected function _normalizeResponseArray($responseArray) {}
}
```

```
<?php
/**
 * @category   Zend
 * @package    Zend_Service_Chargify
 * @link http://docs.chargify.com/api-customers
 */
class Zend_Service_Chargify_Customer extends Zend_Service_Chargify_Abstract
{
  /**
   * (Required)
   *
   * @param string $firstName
   * @return Zend_Service_Chargify_Customer
   */
  public function setFirstName($firstName) {}

  /**
   * (Required)
   *
   * @param string $lastName
   * @return Zend_Service_Chargify_Customer
   */
  public function setLastName($lastName) {}

  /**
   * (Required)
   *
   * @param string $email
   * @return Zend_Service_Chargify_Customer
   */
  public function setEmail($email) {}

  /**
   * (Optional) Company/Organization name
   *
   * @param string $organization
   * @return Zend_Service_Chargify_Customer
   */
  public function setOrganization($organization) {}

  /**
   * (Optional) Phone
   *
   * @param string $phone
   * @return Zend_Service_Chargify_Customer
   */
  public function setPhone($phone) {}

  /**
   * (Optional) Address
   *
   * @param string $address
   * @return Zend_Service_Chargify_Customer
   */
  public function setAddress($address) {}

  /**
   * (Optional) Address2
   *
   * @param string $address
   * @return Zend_Service_Chargify_Customer
```

```
     */
    public function setAddress2($address) {}

    /**
     * (Optional) Country
     *
     * @param string $country
     * @return Zend_Service_Chargify_Customer
     */
    public function setCountry($country) {}

    /**
     * (Optional) State
     *
     * @param string $state
     * @return Zend_Service_Chargify_Customer
     */
    public function setState($state) {}

    /**
     * (Optional) City
     *
     * @param string $city
     * @return Zend_Service_Chargify_Customer
     */
    public function setCity($city) {}

    /**
     * (Optional) Zip
     *
     * @param string $zip
     * @return Zend_Service_Chargify_Customer
     */
    public function setZip($zip) {}

    /**
     * (Optional, but encouraged) The unique identifier used within your own
     * application for this customer
     *
     * @param string|int $reference
     * @return Zend_Service_Chargify_Customer
     */
    public function setReference($reference) {}

    /**
     * The 'page' parameter. Used when listing customers since you can only get 50
     * at a time.
     *
     * @param int $page
     * @return Zend_Service_Chargify_Customer
     */
    public function setPage($page) {}

    /**
     * Create a new customer
     *
     * @return Zend_Service_Chargify_Customer
     */
    public function create() {}

    /**
     * List all customers for a site
     *
     * @return Zend_Service_Chargify_Customer
     * @see Zend_Service_Chargify_Customer::setPage()
     */
    public function listCustomers() {}

    /**
     * Read the customer data for the given Chargify ID
```

```
 *
 * @param int $id
 * @return Zend_Service_Chargify_Customer
 */
public function readByChargifyId($id) {}

/**
 * Read the customer data for the given reference (from your app)
 *
 * @return Zend_Service_Chargify_Customer
 * @see Zend_Service_Chargify_Customer::setReference()
 */
public function readByReference() {}

/**
 * Update the customer record in Chargify.
 *
 * @param int $id
 * @return Zend_Service_Chargify_Customer
 * @see Zend_Service_Chargify_Customer::setFirstName()
 * @see Zend_Service_Chargify_Customer::setLastName()
 * @see Zend_Service_Chargify_Customer::setEmail()
 * @see Zend_Service_Chargify_Customer::setOrganization()
 * @see Zend_Service_Chargify_Customer::setPhone()
 * @see Zend_Service_Chargify_Customer::setAddress()
 * @see Zend_Service_Chargify_Customer::setAddress2()
 * @see Zend_Service_Chargify_Customer::setCity()
 * @see Zend_Service_Chargify_Customer::setState()
 * @see Zend_Service_Chargify_Customer::setZip()
 * @see Zend_Service_Chargify_Customer::setCountry()
 * @see Zend_Service_Chargify_Customer::setReference()
 */
public function update($id) {}

/**
 * When returning multiple products the array is different depending on which
 * format (xml/json) you are using. This normalizes the array for us so we can
 * rely on a consistent structure.
 *
 * @param array $responseArray
 * @return array
```

```
     */
    protected function _normalizeResponseArray($responseArray) {}
  }
```

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