Zend_Service_Emailvision - Nicolas Fabre

<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_Emailvision Component Proposal

Proposed Component Name	Zend_Service_Emailvision
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Service_Emailvision
Proposers	Nicolas Fabre Alexander Veremyev (Zend Liaison)
Revision	0.1 - 1 September 2006: Describe service (wiki revision: 20)

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

Emailvision provide 'Campaign Commander', is an online software application that allows marketers to manage all aspects of their e-mail marketing campaigns including design, planning, delivery, split testing and analytics. The simple-to-use web based interface allows marketers to enjoy advanced features for response optimisation and automation-with minimal technical knowledge and investment.
Zend_Service_Emailvision is a concrete implementation of the emailvision web service

2. References

- Campaign commander Web service Version 2 user guide
- PHP Soap Extension

3. Component Requirements, Constraints, and Acceptance Criteria

- The component must be simple to use.
- The component must be implements allfunctions API
- Handle authentication
- This component will catch errors and exceptions and report them as SOAP faults
- PHP editor code completion should be supported as much as possible under the proposed design.

- Overloading with magic functions (e.g., `__get()`, `__call()`) should be avoided where possible, and all such usage must be documented well for ease of use.

4. Dependencies on Other Framework Components

- Zend_Service_Exception
- PHP extension SOAP

5. Theory of Operation

Zend_Service_Emailvision is a concrete implementation of the emailvision web service

6. Milestones / Tasks

1. Publish design notes
2. Publish proposal based on design notes
3. Arrive on proposal approvable for incubator development
4. Commit working prototype to incubator
5. Commit passing unit tests
6. Write initial documentation
7. Revise code, tests, and docs based on feedback
8. Merge changes with trunk for core release

7. Class Index

- Zend_Service_Emailvision
- Zend_Service_Emailvision_Exception
- Zend_Service_Emailvision_Campaign
- Zend_Service_Emailvision_Campaign_Exception
- Zend_Service_Emailvision_CampaignReport
- Zend_Service_Emailvision_Item
- Zend_Service_Emailvision_MailingList
- Zend_Service_Emailvision_MailingList_Exception
- Zend_Service_Emailvision_Message
- Zend_Service_Emailvision_Message_Exception
- Zend_Service_Emailvision_Segment
- Zend_Service_Emailvision_Segment_Exception
- Zend_Service_Emailvision_SegmentReport
- Zend_Service_Emailvision_TrackdedUrl
- Zend_Service_Emailvision_TrackdedUrl_Exception

8. Use Cases

UC-01

... (see good use cases book)

9. Class Skeletons

```
class Zend_Service_Emailvision_Exception extends Zend_Exception {}

class Zend_Service_Emailvision {
    const API_URI = 'http://emvapi.emv2.com/ccmdws/apiv2?WSDL';
    const API_XSD = 'http://emvapi.emv2.com/ccmdws/apiv2?xsd=1';

    private $_soap;
    private $_connexion;
    private $_authUname;
    private $_authPass;

    /**
     * Constructs a new emailvision Web Services Client
     *
     * @param  string $uname Client username
     * @param  string $pass  Client password
     * @return void
     */
    public function __construct($uname = null, $pass = null){}

    /**
     * Set client username and password
     *
     * @param  string $uname Client user name
     * @param  string $pass  Client password
     * @return Zend_Service_Emailvision Provides a fluent interface
     */
    public function setAuth($uname, $pass){}

    /**
     * Create a new message
     *
     * @param Zend_Service_Emailvision_Message $message
     * @throws Zend_Service_Emailvision_Exception
     */
    public  function createMessage(Zend_Service_Emailvision_Message $message) {}

    /**
     * Delete a message
     *
     * @param int $messageId
     */
    public function deleteMessage($messageId) {}

    /**
     * Get message list
     *
     * @return Zend_Service_Emailvision_Item
     * @throws Zend_Service_Emailvision_Exception
     */
    public function getMessageIds() {}


    /**
     * Get the message details
     *
     * @param int $messageId
     * @return Zend_Service_Emailvision_Message
     */
    public function getMessage($messageId) {}

    /**
     * Update existing message
```

```
     *
     * @param Zend_Service_Emailvision_Message $message
     */
    public function updateMessage(Zend_Service_Emailvision_Message $message) {}

    /**
     * Track all links of the message body
     *
     * @param int $messageId
     */
    public function trackAllLinks($messageId) {}

    /**
     * Create a new segment
     *
     * @param Zend_Service_Emailvision_Segment $segment
     * @return int new segment id
     */
     public function createSegment(Zend_Service_Emailvision_Segment $segment) {}

    /**
     * Get segment list
     *
     *@return Zend_Service_Emailvision_Item
     */
    public function getSegmentIds() {}

    /**
     * Get segment details
     *
     * @param int $segmentId
     * @return Zend_Service_Emailvision_Segment
     */
    public function getSegment($segmentId) {}

    /**
     * Update an existing segment
     *
     * @param Zend_Service_Emailvision_Segment $segment
     */
    public function updateSegment(Zend_Service_Emailvision_Segment $segment) {}

    /**
     * Delete a segment
     *
     * @param int $segmentId
     */
    public function deleteSegment($segmentId) {}

    /**
     * Evaluate Segment (available in version 2.1 only)
     *
     * Evaluating a segment consists in executing the query derives from
     * criteria set against members' database, and count the amount
     * of eligible records.
     *
     * @param int $segmentId
     * @param bool $forceEvaluation
     */
    public function evaluateSegment($segmentId, bool $forceEvaluation) {}

    /**
     * Synchronize segment from a remote system (available in version 2.1 only)
     *
     */
    public function immediateSegment() {}

    /**
     * Create a new mailingList
     *
```

```
     * @param Zend_Service_Emailvision_MailingList $mailingList
     * @return int the new mailingList id value
     */
     public function createMailingList(Zend_Service_Emailvision_MailingList $mailingList) {}

    /**
     * Get mailingList list
     *
     *@return Zend_Service_Emailvision_Item
     */
    public function getMailingListIds() {}

    /**
     * Get mailingList details
     *
     * @param int $mailindListId
     * @return Zend_Service_Emailvision_MailingList
     */
     public function getMailingList($mailindListId) {}

    /**
     * Update an existing mailingList
     *
     * @param Zend_Service_Emailvision_MailingList $mailingList
     */
    public function updateMailingList(Zend_Service_Emailvision_MailingList $mailingList) {}

    /**
     * Delete a mailingList
     *
     * @param int $mailindListId
     */
    public function deleteMailingList($mailindListId) {}

    /**
     * Evaluate mailingList (available in version 2.1 only)
     *
     * Evaluating a mailingList consists in executing the query derives from
     * criteria set against members' database, and count the amount
     * of eligible records.
     *
     * @param int $segmentId
     * @param bool $forceEvaluation
     * @return numeric value, the amount of eligible records, and the last run date
     */
    public function evaluateMailingList($mailingListId, bool $forceEvaluation) {}

    /**
     * Create a new campaing
     *
     * @param Zend_Service_Emailvision_Campaign $campaign
     * @return int the new campaign id value
     */
    public function createCampaign(Zend_Service_Emailvision_Campaign $campaign) {}

    /**
     * Get campaign list
     *
     *@return Zend_Service_Emailvision_Item
     */
    public function getCampaignIds() {}

    /**
     * Get campaign details
     *
     * @param int $campaignId
     * @return Zend_Service_Emailvision_Campaign
     */
    public function getCampaign($campaignId) {}
```

```
    /**
     * Update an existing campaign
     *
     * @param Zend_Service_Emailvision_Campaign $campaign
     */
    public function updateCampaign(Zend_Service_Emailvision_Campaign $campaign) {}

    /**
     * Delete a campaign
     *
     * @param int $campaignId
     */
    public function deleteCampaign($campaignId) {}

    /**
     * Test campaign
     *
     * @param int $campaignId
     * @param array $membersEmail
     * @return int compaign id of the test campaign
     */
    public function testCampaign($campaignId, array $membersEmail) {}

    /**
     * Post campaign
     *
     * @param int $campaignId
     */
    public function postCampaign($campaignId) {}

    /**
     * Request a report by segment
     *
     * @param int $campaignId
     * @param Zend_Service_Emailvision_JobDemand $jobDemand
     */
    public function demandSegmentReports($campaignId, Zend_Service_Emailvision_JobDemand $jobDemand)
{}

    /**
     * Get the status of your report by segment demand
     *
     * @param int $reportId
     * @return string current status of the process
     */
     public function getReportBySegmentStatus($reportId) {}

    /**
     * Retrieve the report by segment results
     *
     * @param int $reportId
     * @param int $numberRow
     * @return list of Zend_Service_Emailvision_SegmentReport
```

```
     */
    public function getReportBySegmentResult($reportId, $numberRow) {}
}
```

```
class Zend_Service_Emailvision_Campaign_Exception extends  Zend_Service_Emailvision_Exception {}

class Zend_Service_Emailvision_Campaign {

    /**
     * Constructs a new campaign
     *
     * @param array $campaign
     * @return void
     */
    public function __construct(array $campaign) {}

    /**
     * Magic function so that $obj->value will work.
     *
     * @param string $name
     * @param mixed $value
     */
    public function __set($name, $value) {}

    /**
     * Magic function so that $obj->value will work.
     *
     * @param string $name
     * @return mixed
     */
    public function __get($name) {}

}
```

```
class Zend_Service_Emailvision_Item implements Iterator, Countable {

    /**
     * Magic function so that $obj->value will work.
     *
     * @param string $name
     * @param mixed $value
     */
    public function __set($name, $value){}

    /**
     * Magic function so that $obj->value will work.
     *
     * @param string $name
     * @return mixed
     */
    public function __get($name){}

    /**
     * Return an associative array of the stored data.
     *
     * @return array
     */
    public function toArray(){}

    /**
     * Support isset() overloading on PHP 5.1
     *
     * @param string $name
```

```
     * @return boolean
     */
    protected function __isset($name){}

    /**
     * Defined by Countable interface
     *
     * @return int
     */
    public function count(){}

    /**
     * Defined by Iterator interface
     *
     * @return mixed
     */
    public function current(){}

    /**
     * Defined by Iterator interface
     *
     * @return mixed
     */
    public function key(){}

    /**
     * Defined by Iterator interface
     *
     */
    public function next(){}

    /**
     * Defined by Iterator interface
     *
     */
    public function rewind(){}

    /**
     * Defined by Iterator interface
     *
     * @return boolean
```

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

```
class Zend_Service_Emailvision_CampaignReport {

}
```

```
class Zend_Service_Emailvision_MailingList_Exception extends Zend_Service_Emailvision_Exception {}

class Zend_Service_Emailvision_MailingList {

}
```

```
class Zend_Service_Emailvision_Message_Exception extends Zend_Service_Emailvision_Exception {}

class Zend_Service_Emailvision_Message {
    const EMV_BODY = '[...]';
    /**
     * Constructs a message with $message should be an associative
     * array with one or more of the following keys (name key is mandatory):
     * - body: html message
     * - bounceBackFlag: 0 or 1 to use the message as an automatic answer for futur campaings
     * - description: description of the contents of the message
     * - encoding: character encoding
     * - from:
     * - id: message id
     * - name: name of the message for storage
     * - passThroughFlag: 0 or 1 if you want to ignore Mailing Frequency Limit
     * - subject: title message
     * - to:
     * - trakedUrls: collection of Zend_Service_Emailvision_TrackedUrl
     * - type: EMAIL or SMS
     * @param array $message
     * @return void
     */
    public function __construct(array $message) {}

    /**
     * Magic function so that $obj->value will work.
     *
     * @param string $name
     * @param mixed $value
     */
    public function __set($name, $value) {}

    /**
     * Magic function so that $obj->value will work.
     *
     * @param string $name
     * @return mixed
     */
    public function __get($name) {}
}
```

```
class Zend_Service_Emailvision_Segment_Exception extends Zend_Service_Emailvision_Exception {}

class Zend_Service_Emailvision_Segment {}
```

```
class Zend_Service_Emailvision_SegmentReport { }
```

```
class Zend_Service_Emailvision_TrackedUrl_Exception extends Zend_Service_Emailvision_Exception {}

class Zend_Service_Emailvision_TrackedUrl implements Iterator, Countable {
    /**
     * Constructs a trackedUrl with $trackedUrl should be an associative
     * array with one or more of the following keys (type, name, order keys are mandatory):
     * - type: STANDARD, PERSONALIZABLE, UPDATE, ACTION, UNJOIN, MIRROR, FEED, RSSITEM
     * - url: url tracked
     * - name: title Url
     * - order: start at 1
     * - action: only for action tracked url
     * - messageIdKo: id of the reponse message when the process is done
     * - messageIdOk: id of the reponse message when the process fails
     * - pageKo: url of the page to display when the process is done
     * - pageOk: url of the page to display when the process fails
     * @param array $trackedUrl
     * @return void
     */
    public function __construct(array $trackedUrl) {}

    /**
     * Magic function so that $obj->value will work.
     *
     * @param string $name
     * @param mixed $value
     */
    public function __set($name, $value) {}

    /**
     * Magic function so that $obj->value will work.
     *
     * @param string $name
     * @return mixed
     */
    public function __get($name) {}

    /**
     * Return an associative array of the stored data.
     *
     * @return array
     */
    public function toArray(){}

    /**
     * Support isset() overloading on PHP 5.1
     *
     * @param string $name
     * @return boolean
     */
    protected function __isset($name){}

    /**
     * Defined by Countable interface
     *
     * @return int
     */
    public function count(){}

    /**
     * Defined by Iterator interface
     *
     * @return mixed
     */
```

```
    public function current(){}

    /**
     * Defined by Iterator interface
     *
     * @return mixed
     */
    public function key(){}

    /**
     * Defined by Iterator interface
     *
     */
    public function next(){}

    /**
     * Defined by Iterator interface
     *
     */
    public function rewind(){}

    /**
     * Defined by Iterator interface
     *
     * @return boolean
```

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

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