# Zend_File_Transfer - Thomas Weidner

<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_File_Transfer Component Proposal

| | |
|---|---|
| **Proposed Component Name** | Zend_File_Transfer |
| **Developer Notes** | http://framework.zend.com/wiki/display/ZFDEV/Zend_File_Transfer |
| **Proposers** | Thomas Weidner<br>Matthew Weier O'Phinney (Zend Liaison) |
| **Revision** | 1.0 - 27 Apr 2007: Initial release<br>1.1 - 03 May 2007: Small reworks to the API<br>1.2 - 05 May 2007: Added hook for check as discussed<br>2.0 - 06 Aug 2007: Renamed to Zend_Http_Upload<br>3.0 - 22 Dec 2007: Renamed to Zend_File_Uploader and completly reworked the API<br>4.0 - 20 Jan 2008: Renamed to Zend_File_Transfer, API reworked (wiki revision: 28) |

## Table of Contents



## 1. Overview

Zend_File_Transfer is a component to handle File Up- and Downloads within the Zend Framework in a standardized way.

## 2. References

- PHP Manual file upload
- Multiple File uploads
- File Upload Progress Meter
- PHP Freaks

## 3. Component Requirements, Constraints, and Acceptance Criteria

No requirements...

But to receive upload-processing informations a pecl extension and minimum PHP 5.2 is needed.
Otherwise the class can not provide the processing information.

## 4. Dependencies on Other Framework Components

- Zend_Exception

## 5. Theory of Operation

This component is meant to handle file up- and downloads within the Zend Framework. Is should provide an simple and generic way for all users.
Several tasks are possible:

- Upload multiple files
- Set validators
- Set filters
- Set paths
- Rename files
- Provide a way to get the download progress data for the handled files (could be limited to 5.2)
- Wrapper for downloads
- Different adapters for HTTP POST, WEBDAV, FTP, AJAX and so on...

## 6. Milestones / Tasks

- Milestone 1: [FINISHED] Proposal finished
- Milestone 2: [FINISHED] Proposal accepted
- Milestone 3: Working release
- Milestone 4: Unit tests
- Milestone 5: Documentation
- Milestone 6: Future enhancements

## 7. Class Index

- Zend_File_Transfer_Exception
- Zend_File_Transfer
- Zend_File_Transfer_Protocol
- Zend_File_Transfer_Protocol_Http
- Zend_File_Transfer_Procotol_Ftp
- Zend_File_Transfer_Procotol_WebDav
- Zend_File_Transfer_Procotol_Ajax

## 8. Use Cases

**Upload defined file**

Default for transfer is http download. This code shows the simplest way to have downloads integrated.

```
$files = new Zend_File_Transfer();
$files->addFiles($field1);
try {
    $files->receive('C:\path\to\put\files');
} catch (Zend_File_Transfer_Exception $ex) {
    // Files not uploaded or other problems
}
```

### Set validators

Validators can be used to check several options of files which are loaded. Validators could possibly be Zend_Validate validators... but this would be checked at time of integration. Not sure if this would fit all needs. But Zend_File_Transfer will integrate several own validators like FileSize, FileExtension, MimeType and so on...

```
$files = new Zend_File_Transfer();
$files->addFiles($field1);
$files->addValidator('MaxSize', 2000);
$files->addValidator('MimeType', 'gif'); // mime type
if (!$files->isValid()) {
    throw new Exception("File does not fit our needs!!");
}
try {
    $files->receive('C:\path\to\put\files');
} catch (Zend_File_Transfer_Exception $ex) {
    // Files not uploaded or other problems
}
```

### Validators for single files/fields

Validators can also be set for single files, or for all. Also isUploaded can check for all or single files. Of course the API supports fluid interface where applicable.

```
$files = new Zend_File_Transfer();
$files->addFiles($field1)
      ->addFiles($field2)
      ->addFiles($field, array('MinSize' => 2000, 'Extension' => 'jpg')
      ->addValidator('MaxSize', 2000);
if (!$files->isUploaded()) {
    print "not all files uploaded";
}
if (!$files->isUploaded($field1)) {
    print "file ".$field1." was not uploaded";
}
if (!$files->isValid($field1)) {
    // validation failed
}
```

### Change content with filters

With filters it is possible to change content of downloaded files before they are stored. Doublicate transfered files to a second directory, or change content with self written filters in this example change linebreaks of textfiles from unix to windows. Several filters are thinkable.

```
$files = new Zend_File_Transfer();
$files->setValidator(array('Extension' => 'jpg|pdf', 'MaxSize' => '20MB'))
      ->addFilter('Copy', '/home/myspace')   // produces a copy to this directory
      ->addFilter('Copy', '/home/yourspace') // We would have the files now 3 times
      ->addFilter('LineBreakUnixToWindows')
      ->receive();
```

### HTTP PUT Uploads

Also other adapters can be used. In our example http put, but also other adapters like FTP, WEBDAV or AJAX will be integrated. Within all adapters the API will be the same.

```
try {
    $files = new Zend_File_Transfer('HTTP_PUT', 'C:\temp\uploads') // set generic
destination
    ->addFiles($field)
    ->setValidator('MaxSize', 2000)
    ->receive();
} catch (Zend_File_Transfer_Exception $e) {
    print "Upload failed:" . $e->getMessage();
}
```

### FTP Downloads

The API can also be used for downloads. It will work as wrapper so the user does not see where the original files is located and ZF will send the file to the user.

```
try {
    $files = new Zend_File_Transfer('FTP', array('user' => 'adam', 'pwd' => 'sandler',
'server' => 'ftp.myserver.com/public'));
    $files->addFiles('*.*')
          ->setValidator('MaxSize', 2000)
          ->send();
} catch (Zend_File_Transfer_Exception $e) {
    print "HTTP 500: " . $e->getMessage();
}
```

### WebDav Uploads

All adapters use the same API... here WebDav

```
try {
    $files = new Zend_File_Transfer('WEBDAV', array('user' => 'adam', 'pwd' =>
'sandler', 'server' => 'myserver.com'));
    $files->addFiles('*.*')
          ->setValidator('MaxSize', 2000)
          ->receive();
} catch (Zend_File_Transfer_Exception $e) {
    print "Download was not successfull";
}
```

### Ajax Uploads

There is also an idea of creating an Ajax adapter which would make it possible to upload files in a form while the user has not to wait until the upload is finished and can work on the form or make other things while the file is uploaded in background with ajax. Also the status (amount of downloaded files, progress and so on) is avaiable to the user while the upload is in progress and can be displayed to him as additional information.

Actually there is no code because this will also have to be integrated in the form or the view the user get's displayed.

## 9. Class Skeletons

```
class Zend_File_Transfer_Exception extends Zend_Exception {}

class Zend_File_Transfer {

    /**
     * Constructor
     * @param String|Zend_File_Transfer_Protocol $protocol  - OPTIONAL protocol object
to use
     * @param Array|String List                 $options   - OPTIONAL options for
this adapter
     */
    public function __construct($protocol = null, array $options = null) {}

    /**
     * Add files to process
     * @param String|Array $file     - File/s to process
     * @param Array|Object $validator - OPTIONAL Array of validators to set for all of
these files
     */
    public function addFiles($file, array $validator = Array()) {}

    /**
     * Add filetypes to process
     * @param String|Array $type      - Filetype/s to process
     * @param Array|Object $validator - OPTIONAL Array of validators to set for all of
these files
     */
    public function addFileType($filetype, array $validator = Array()) {}

    /**
```

```
     * Set validators, deleting existing ones
     * @param String|Array|Zend_File_Transfer_Validator  $validator - Validator to set
     * @param              $option - OPTIONAL Option for this validator
     * @param String|Array $file   - OPTIONAL File/s for which to set the maximum
size, if not set for all in sum
     */
    public function setValidator($validator, $option = null, $file = null) {}

    /**
     * Adds new validators to the chain
     * @param String|Array|Zend_File_Transfer_Validator  $validator - Validator to set
     * @param              $option - OPTIONAL Option for this validator
     * @param String|Array $file   - OPTIONAL File/s for which to set the maximum
size, if not set for all in sum
     */
    public function addValidator($validator, $option = null, $file = null) {}

    /**
     * Set filters, overwriting existing ones
     * @param String|Array|Zend_File_Transfer_Filter  $filter - Filter to set
     * @param              $option - OPTIONAL Options for this filter
     * @param String|Array $file   - OPTIONAL File/s for which to set the maximum
size, if not set for all in sum
     */
    public function setFilter($filter, $option = null, $file = null) {}

    /**
     * Adds new filters to the chain
     * @param String|Array|Zend_File_Transfer_Filter  $filter - Filter to set
     * @param              $option - OPTIONAL Options for this filter
     * @param String|Array $file   - OPTIONAL File/s for which to set the maximum
size, if not set for all in sum
     */
    public function addFilter($filter, $option = null, $file = null) {}

    /**
     * Set new location for files, can be set per single file
     * @param String       $destination - Where to store the file/s
     * @param String|Array $files       - OPTIONAL File/s for which to set the
destination
     *                                    , if not set for all in sum
     */
    public function setDestination($destination, $files = null) {}

    /**
     * Receives files from the user (upload to server), move, copy whatever was
defined...
     * @param String       $option    - OPTIONAL Option for this validator
     * @param String|Array $files     - OPTIONAL Files to process
     * @throws Zend_Http_Transfer_Exception
     */
    public function receive($option = null, $files = null) {}

    /**
     * Sends files to the user (download from server), move, copy whatever was
defined...
     * @param String       $option    - OPTIONAL Option for this validator
     * @param String|Array $files     - OPTIONAL Files to process
     * @throws Zend_Http_Transfer_Exception
```

```
     */
    public function send($option = null, $files = null) {}

    /**
     * Validates related to given parameters
     *
     * @param String       $option    - OPTIONAL Option for this validator
     * @param String|Array $file      - OPTIONAL File/s to validate
     * @throws Zend_Http_Transfer_Exception
     */
    public function isValid($option = null, $file = null) {}

    /**
     * Check if all or the given files were uploaded
     * @param String|Array $file - OPTIONAL File/s to check if they were uploaded
     */
    public function isUploaded($file = null) {}

    /**
     * Get actual upload progress, works only for php 5.2 with installed pecl
extension!
     */
    public function getProgress() {}

    /**
     * Returns all files to upload
     */
    public function getFiles() {}

    /**
     * Returns all set validators
     */
    public function getValidator() {}

    /**
     * Returns all set filters
     */
    public function getFilter() {}

}

abstract class Zend_File_Transfer_Protocol

class Zend_File_Transfer_Protocol_Http_Post
class Zend_File_Transfer_Protocol_Http_Put
class Zend_File_Transfer_Protocol_Ftp
class Zend_File_Transfer_Protocol_WebDav
```

```
class Zend_File_Transfer_Protocol_Ajax
```

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