# Zend_TimeSync - Thomas Weidner & Andries Seutens

<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_TimeSync Component Proposal

| | |
|---|---|
| **Proposed Component Name** | Zend_TimeSync |
| **Developer Notes** | http://framework.zend.com/wiki/display/ZFDEV/Zend_TimeSync |
| **Proposers** | • Thomas Weidner<br>  • Team lead and author<br>• Andries Seutens<br>  • Author |
| **Revision** | 5.0 - 2 December 2007: Finished implementation<br>4.0 - 28 November 2006: Reworked proposal based on actual work and feedback from Zend<br>3.0 - 06 November 2006: Reworked proposal based on feedback from various people<br>2.0 - 06 November 2006: Reworked proposal based on actual work<br>1.0 - 11 October 2006: Proposal submitted (wiki revision: 25) |

## 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

Zend_TimeSync implements the ability for TimeSyncronisation to the Framework.

Note that best practices would rely on a local service to sync the local clock to remote NTP servers, so that microtime() would simply return precisely accurate results. However, sometimes a developer must use a server lacking this kind of local service.

## 2. References

- NTP
- SNTP
- NTP Server Pool for public use

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

- Zend_TimeSync needs Zend_Date to handle timestamps properly
- Zend_TimeSync can not change the server's time
- Zend_TimeSync will use an internal caching mechanism (Zend_Cache) to prevent overload

## 4. Dependencies on Other Framework Components

- Zend
- Zend_Date
- Zend_Cache
- Zend_Exception

## 5. Theory of Operation

This component is able to receive internet or network time from a timeserver using the NTP or SNTP protocol. By using this component a page/script is able to act indepentendly from the timesettings of the server the script is running. However, in practical terms, this component uses internaly a caching mechanism (Zend_Cache) to prevent overload.

Zend_TimeSync will allow multiple servers to be added to the class constructor as an array. These servers would then be used as "fallback" servers. The class will automaticly switch to the next timeserver - which can be a mix of ntp and sntp timeservers - on port/server error.

This class is not able to change the server's time but it will return a date object from which the difference to the servers time can be worked with.

## 6. Milestones / Tasks

- Milestone 1: [DONE] proposal written
- Milestone 2: [DONE] implementing Ntp
- Milestone 3: [DONE] implementing Sntp
- Milestone 4: [DONE] Unit tests and debugging
- Milestone 5: [DONE] Documentation

## 7. Class Index

- Zend_TimeSync
- Zend_TimeSync_Protocol
- Zend_TimeSync_Ntp
- Zend_TimeSync_Sntp
- Zend_TimeSync_Exception

## 8. Use Cases

Getting actual time from a **NTP** server with a standard port:

```
$server = new Zend_TimeSync('ntp://time.ntp.org');
$result = $server->getDate();

echo $result->getIso();
```

Getting actual time from a **SNTP** server with standard port:

```
$server = new Zend_TimeSync('sntp://time.sntp.org');
$result = $server->getDate();

echo $result->getIso();
```

Getting actual time from an **NTP** server with standard port, and a list of **NTP** fallback servers provided (remember, first in == first out)

```
$serverlist = array('ntp://time.ntp.org',
                    'ntp://time2.ntp.org',
                    'ntp://time3.ntp.org');

$server = new Zend_TimeSync($serverlist);
$result = $server->getDate();

echo $result->getIso();
```

Getting actual time from a **SNTP** server with standard port, and a list of **SNTP** fallback servers provided (remember, first in == first out)

```
$serverlist = array('sntp://time.sntp.org',
                    'sntp://time2.sntp.org',
                    'sntp://time3.sntp.org');

$server = new Zend_TimeSync($serverlist);
$result = $server->getDate();

echo $result->getIso();
```

Getting actual time from an **NTP** server with a none standard port, and with a list of **NTP** fallback servers, which also run on a none standard port (remember, first in == first out)

```
$serverlist = array('ntp://time.ntp.org:2202',
                    'ntp://time2.ntp.org:2202',
                    'ntp://time3.ntp.org:2202');

$server = new Zend_TimeSync($serverlist);
$result = $server->getDate();

echo $result->getIso();
```

Getting actual time from an **SNTP** server with a none standard port, and with a list of **SNTP** fallback servers, which also run on a none standard port (remember, first in == first out)

```
$serverlist = array('sntp://time.sntp.org:2202',
                    'sntp://time2.sntp.org:2202',
                    'sntp://time3.sntp.org:2202');

$server = new Zend_TimeSync($serverlist);
$result = $server->getDate();

echo $result->getIso();
```

Getting actual time from an **NTP** server with a standard port, and with a list of **SNTP** fallback servers running on a none standard port(remember, first in == first out)

```
$serverlist = array('ntp://time.ntp.org',
                    'sntp://time2.sntp.org:2202',
                    'sntp://time3.sntp.org:2202');

$server = new Zend_TimeSync($serverlist);
$result = $server->getDate();

echo $result->getIso();
```

taking care of errors/exceptions:

```
$example = array(
    'sntp://bdqdqe.pool.ntp.org',
    'sntp://time-C.timefreq.bldrdoc.gov',
    'sntp://time-test-three.be',
    'sntp://time.windows.com'
);

$server = new Zend_TimeSync($example);

// we want to query the 3th server and skip the fallback mechanism

$invalidServer = $server->get(2);

// this will throw the exception because the 3th server is invalid

try {
    $foo = $invalidServer->getDate();
    echo $foo->getIso());
} catch (Zend_TimeSyncProtocolException $e) {
    echo $e->getMessage();
}

// use the fallback mechanism and return the result from
// the first server that returns a valid result:

try {
    $result = $server->getDate();

    echo $result->getIso(); // sntp://time-C.timefreq.bldrdoc.gov

} catch (Zend_TimeSync_Exception $e) {

    $exceptions = $e->get();

    foreach ($exceptions as $key => $myException) {
        echo $myException->getMessage();
        echo '<br />';
    }
}
```

## 9. Class Skeletons

Zend_Timesync returnes a Zend_Date object which represents the time from the (S)NTP Server

```
class Zend_TimeSync implements IteratorAggregate
{
    /**
     * Zend_TimeSync constructor.
     *
     * The constructor takes one to two parameters.
     *
     * The first parameter is $server, which may be a single string
     * representation for a timeserver, or a structured array for
```

```
     * multiple timeservers.
     *
     * Each server must be provided with a valid scheme, and may
     * contain an optional port number. If no port number has been
     * suplied, the default matching port number will be used.
     *
     * Valid schemes are:
     * - ntp
     * - sntp
     *
     * The second parameter is $options, and it is optional. If not
     * specified, default options will be used.
     *
     * @param mixed $server
     * @param array $options
     * @return Zend_TimeSync
     */
    public function __construct($server, $options = array())
    {}

    /**
     * Sets a single option. It replaces any currently defined.
     *
     * @param mixed $key
     * @param mixed $value
     * @throws Zend_TimeSync_Exception
     */
    public function setOption($key, $value)
    {}

    /**
     * Set options. These replace any currently defined.
     *
     * @param array $options
     * @throws Zend_TimeSync_Exception
     */
    public function setOptions($options = array())
    {}

    /**
     * Mark a nameserver as current.
     *
     * @param integer $flag
     * @throws Zend_TimeSync_Exception
     */
    public function setCurrent($flag)
    {}

    /**
     * Returns an array of options that have been set.
     *
     * @return array
     */
    public function getOptions()
    {}

    /**
     * Returns the value to the option, if any.
     * If the option was not found, this function returns false.
```

```
     *
     * @param integer $flag
     * @return mixed
     * @throws Zend_TimeSync_Exception
     */
    public function getOption($flag)
    {}

    /**
     * Receive a specified timeservers
     *
     * @param integer $flag
     * @return object
     * @throws Zend_TimeSync_Exception
     */
    public function get($flag)
    {}

    /**
     * Receive the timeserver that is currently set
     *
     * @return object
     * @throws Zend_TimeSync_Exception
     */
    public function getCurrent()
    {}

    /**
     * Add a timeserver to the server list
     *
     * @param $server server name including scheme specification
     * @return void
     * @throws Zend_TimeSync_Exception
     */
    public function addServer($server)
    {}

    /**
     * Query the timeserver list using the fallback mechanism
     *
     * @param $locale optional locale
     * @return object
     * @throws Zend_TimeSync_Exception
     */
    public function getDate($locale = false)
    {}
}

class Zend_TimeSync_Protocol
{
    /**
     * Connect to the timeserver. If called when the socket is
     * already connected, it disconnects and connects again.
     *
     * @access protected
     * @return boolean
     */
    protected function _connect()
    {}
```

```
    /**
     * Disconnects from the peer, closes the socket.
     *
     * @access protected
     * @return bool
     */
    protected function _disconnect()
    {}

    /**
     * Query this timeserver
     *
     * @param $locale optional locale
     * @return object
     * @throws Zend_TimeSync_Exception
     */
    public function getDate($locale = false)
    {}
}

class Zend_TimeSync_Ntp extends Zend_TimeSync_Protocol {}

class Zend_TimeSync_Sntp extends Zend_TimeSync_Protocol {}

class Zend_TimeSync_Exception extends Zend_Exception
{
    /**
     * Add an internal exception
     *
     * @param $exception
     */
     function add(Zend_TimeSync_Exception $exception)
```

```
        {}
  }
```

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