Исправление проблемы с транслитерацией

В связи с прекращением поддержки протокола Яндекс.Translate 1.0 режим перевода/транслитерации в версиях HostCMS до 6.5.1 прекратил работу.
Для поддержкит протокола Яндекс.Translate 1.5 необходимо:

1. В константу **YANDEX_TRANSLATE_KEY** внести ключ, получить который можно на странице https://tech.yandex.ru/key/form.xml?service=trnsl
2. Заменить метод **core_str::translate**

```
    /**
     * Translation from russian to english
     * @param string $string source string
     * @return string
     */
    static public function translate($string)
    {
        if (defined('YANDEX_TRANSLATE_KEY') &&
strlen(YANDEX_TRANSLATE_KEY))
        {
            $url =
'https://translate.yandex.net/api/v1.5/tr.json/translate?' .
                'key=' . urlencode(YANDEX_TRANSLATE_KEY) .
                '&text=' . urlencode($string) .
                '&lang=en&format=plain';

            $Core_Http = Core_Http::instance()
                ->url($url)
                ->timeout(3)
                ->execute();

            $data = trim($Core_Http->getBody());

            if (strlen($data))
            {
                $oData = json_decode($data);

                if (is_object($oData) && $oData->code == 200 &&
isset($oData->text[]))
                {
                    return $oData->text[];
                }
            }
        }
        /*else
        {
            Core_Log::instance()->clear()
                ->status(Core_Log::$MESSAGE)
                ->write('Can not translate. Constant
```

```
YANDEX_TRANSLATE_KEY is undefined.');
        }*/

        return NULL;
    }
```

3. Заменить метод **core_http::execute**

```
/**
     * Executes the business logic.
     */
    public function execute()
    {
        $aUrl = @parse_url(trim($this->_url));

        $scheme = strtolower(Core_Array::get($aUrl, 'scheme', 'http'));

        // Change https port
        $scheme == 'https' && $this->_port == 80
            && $this->_port = 443;

        $path = isset($aUrl['host']) && isset($aUrl['path'])
            ? $aUrl['path']
            : '/';

        $host = Core_Array::get($aUrl, 'host', '');

        $query = isset($aUrl['query']) ?
            '?' . $aUrl['query']
            : '';

        $this->_referer = is_null($this->_referer)
            ? "{$scheme}://{$host}"
            : $this->_referer;

        return $this->_execute($host, $path, $query);
    }
```

4. Заменить класс (на некоторых версиях не нужно) **core_http_socket**

```
<?php

defined('HOSTCMS') || exit('HostCMS: access denied.');

/**
 * Http socket driver
 *
 * @package HostCMS 6\Core\Http
 * @version 6.x
 * @author Hostmake LLC
 * @copyright © 2005-2015 ООО "Хостмэйк" (Hostmake LLC),
```

```
http://www.hostcms.ru
 */
class Core_Http_Socket extends Core_Http
{
    /**
     * Send request
     * @param string $host host
     * @param string $path path
     * @param string $query query
     * @return self
     */
    protected function _execute($host, $path, $query)
    {
        // для 443 порта в  fsockopen  перед хостом нужно добавлять  ssl://
        $socketHost = $this->_port == 443
            ? 'ssl://' . $host
            : $host;

        if (!function_exists('fsockopen'))
        {
            throw new Core_Exception("Fsockopen has been disabled,
please contact your system administrator!");
        }

        $fp = @fsockopen($socketHost, $this->_port, $errno, $errstr,
$this->_timeout);

        if (!$fp)
        {
            throw new Core_Exception("Fsockopen failed '%errstr'
(%errno)",
                array('%errstr' => $errstr, '%errno' => $errno), $errno
            );
        }

        $out = "{$this->_method} {$path}{$query} HTTP/1.0\r\n";
        $out .= "Content-Type: {$this->_contentType}\r\n";
        $out .= "Referer: {$this->_referer}\r\n";
        $out .= "User-Agent: {$this->_userAgent}\r\n";

        // Additional headers
        foreach ($this->_additionalHeaders as $name => $value)
        {
            $out .= "{$name}: {$value}\r\n";
        }

        $out .= "Host: {$host}\r\n";

        $bIsPost = $this->_method != 'GET' && ($this->_rawData ||
count($this->_data) > );
```

```
        if ($bIsPost)
        {
            if ($this->_rawData)
            {
                $sPost = $this->_rawData;
            }
            else
            {
                $aData = array();
                foreach ($this->_data as $key => $value)
                {
                    $aData[] = urlencode($key) . '=' .
urlencode($value);
                }

                $sPost = implode('&', $aData);
            }

            $out .= "Content-length: " . strlen($sPost) . "\r\n";
        }

        $out .= "Connection: Close\r\n\r\n";

        if ($bIsPost)
        {
            $out .= $sPost . "\r\n\r\n";
        }

        fwrite($fp, $out);

        if (function_exists('stream_set_timeout'))
        {
            stream_set_timeout($fp, $this->_timeout);
        }

        $datastr = '';

        while (!feof($fp))
        {
            $datastr .= fgets($fp, 65536);
        }

        fclose($fp);

        $aTmp = explode("\r\n\r\n", $datastr, 2);
        unset ($datastr);

        $this->_headers = Core_Array::get($aTmp, );
        $this->_body = Core_Array::get($aTmp, 1);

        return $this;
```

```
    }
}
```