```php
<?php
/**
 * @package     Joomla.Platform
 * @subpackage  HTML
 *
 * @copyright   Copyright (C) 2005 - 2012 Open Source Matters, Inc. All rights reserved.
 * @license     GNU General Public License version 2 or later; see LICENSE
 */

defined('JPATH_PLATFORM') or die;

// Register the element class with the loader.
JLoader::register('JElement', dirname(__FILE__) . '/parameter/element.php');

/**
 * Parameter handler
 *
 * @package     Joomla.Platform
 * @subpackage  Parameter
 * @since       11.1
 * @deprecated  12.1  Use JForm instead
 */
class JParameter extends JRegistry
{
    /**
     * @var    string  The raw params string
     * @since  11.1
     */
    protected $_raw = null;

    /**
     * @var    object  The XML params element
     * @since  11.1
     */
    protected $_xml = null;

    /**
     * @var    array  Loaded elements
     * @since  11.1
     */
    protected $_elements = array();

    /**
     * @var    array  Directories, where element types can be stored
     * @since  11.1
     */
    protected $_elementPath = array();

    /**
     * Constructor
     *
     * @param   string  $data  The raw parms text.
     * @param   string  $path  Path to the XML setup file.
     *
     * @deprecated  12.1
     * @since   11.1
     */
    public function __construct($data = '', $path = '')
    {
        // Deprecation warning.
        JLog::add('JParameter::__construct is deprecated.', JLog::WARNING, 'deprecated');

        parent::__construct('_default');

        // Set base path.
        $this->_elementPath[] = dirname(__FILE__) . '/parameter/element';

        if ($data = trim($data))
        {
            if (strpos($data, '{') === 0)
            {
                $this->loadString($data);
            }
            else
            {
                $this->loadINI($data);
            }
        }

        if ($path)
```

```
81          {
82              $this->loadSetupFile($path);
83          }
84
85          $this->_raw = $data;
86      }
87
88      /**
89       * Sets a default value if not alreay assigned.
90       *
91       * @param   string  $key      The name of the parameter.
92       * @param   string  $default  An optional value for the parameter.
93       * @param   string  $group    An optional group for the parameter.
94       *
95       * @return  string  The value set, or the default if the value was not previously set (or null).
96       *
97       * @deprecated  12.1
98       * @since   11.1
99       */
100     public function def($key, $default = '', $group = '_default')
101     {
102         // Deprecation warning.
103         JLog::add('JParameter::def is deprecated.', JLog::WARNING, 'deprecated');
104
105         $value = $this->get($key, (string) $default, $group);
106
107         return $this->set($key, $value);
108     }
109
110     /**
111      * Sets the XML object from custom XML files.
112      *
113      * @param   JSimpleXMLElement  &$xml  An XML object.
114      *
115      * @return  void
116      *
117      * @deprecated  12.1
118      * @since   11.1
119      */
120     public function setXML(&$xml)
121     {
122
123         // Deprecation warning.
124         JLog::add('JParameter::setXML is deprecated.', JLog::WARNING, 'deprecated');
125
126         if (is_object($xml))
127         {
128             if ($group = $xml->attributes('group'))
129             {
130                 $this->_xml[$group] = $xml;
131             }
132             else
133             {
134                 $this->_xml['_default'] = $xml;
135             }
136
137             if ($dir = $xml->attributes('addpath'))
138             {
139                 $this->addElementPath(JPATH_ROOT . str_replace('/', DS, $dir));
140             }
141         }
142     }
143
144     /**
145      * Bind data to the parameter.
146      *
147      * @param   mixed   $data   An array or object.
148      * @param   string  $group  An optional group that the data should bind to. The default group is used if not supplied.
149      *
150      * @return  boolean  True if the data was successfully bound, false otherwise.
151      *
152      * @deprecated  12.1
153      * @since   11.1
154      */
155     public function bind($data, $group = '_default')
156     {
157         // Deprecation warning.
158         JLog::add('JParameter::bind is deprecated.', JLog::WARNING, 'deprecated');
159
160         if (is_array($data))
```

```
161         {
162
163             return $this->loadArray($data);
164         }
165         elseif (is_object($data))
166         {
167             return $this->loadObject($data);
168         }
169         else
170         {
171             return $this->loadString($data);
172         }
173     }
174
175     /**
176      * Render the form control.
177      *
178      * @param   string  $name   An optional name of the HTML form control. The default is 'params' if not supplied.
179      * @param   string  $group  An optional group to render.  The default group is used if not supplied.
180      *
181      * @return  string  HTML
182      *
183      * @deprecated  12.1
184      * @since   11.1
185      */
186     public function render($name = 'params', $group = '_default')
187     {
188         // Deprecation warning.
189         JLog::add('JParameter::render is deprecated.', JLog::WARNING, 'deprecated');
190
191         if (!isset($this->_xml[$group]))
192         {
193             return false;
194         }
195
196         $params = $this->getParams($name, $group);
197         $html = array();
198
199         if ($description = $this->_xml[$group]->attributes('description'))
200         {
201             // Add the params description to the display
202             $desc = JText::_($description);
203             $html[] = '<p class="paramrow_desc">' . $desc . '</p>';
204         }
205
206         foreach ($params as $param)
207         {
208             if ($param[0])
209             {
210                 $html[] = $param[0];
211                 $html[] = $param[1];
212             }
213             else
214             {
215                 $html[] = $param[1];
216             }
217         }
218
219         if (count($params) < 1)
220         {
221             $html[] = "<p class=\"noparams\">" . JText::_('JLIB_HTML_NO_PARAMETERS_FOR_THIS_ITEM') . "</p>";
222         }
223
224         return implode(PHP_EOL, $html);
225     }
226
227     /**
228      * Render all parameters to an array.
229      *
230      * @param   string  $name   An optional name of the HTML form control. The default is 'params' if not supplied.
231      * @param   string  $group  An optional group to render.  The default group is used if not supplied.
232      *
233      * @return  array
234      *
235      * @deprecated  12.1
236      * @since   11.1
237      */
238     public function renderToArray($name = 'params', $group = '_default')
239     {
240
```

```
241         // Deprecation warning.
242         JLog::add('JParameter::renderToArray is deprecated.', JLog::WARNING, 'deprecated');
243
244         if (!isset($this->_xml[$group]))
245         {
246             return false;
247         }
248         $results = array();
249         foreach ($this->_xml[$group]->children() as $param)
250         {
251             $result = $this->getParam($param, $name, $group);
252             $results[$result[5]] = $result;
253         }
254         return $results;
255     }
256
257     /**
258      * Return the number of parameters in a group.
259      *
260      * @param   string  $group  An optional group. The default group is used if not supplied.
261      *
262      * @return  mixed  False if no params exist or integer number of parameters that exist.
263      *
264      * @deprecated  12.1
265      * @since   11.1
266      */
267     public function getNumParams($group = '_default')
268     {
269         // Deprecation warning.
270         JLog::add('JParameter::getNumParams is deprecated.', JLog::WARNING, 'deprecated');
271
272         if (!isset($this->_xml[$group]) || !count($this->_xml[$group]->children()))
273         {
274             return false;
275         }
276         else
277         {
278             return count($this->_xml[$group]->children());
279         }
280     }
281
282     /**
283      * Get the number of params in each group.
284      *
285      * @return  array  Array of all group names as key and parameters count as value.
286      *
287      * @deprecated  12.1
288      * @since   11.1
289      */
290     public function getGroups()
291     {
292         // Deprecation warning.
293         JLog::add('JParameter::getGroups is deprecated.', JLog::WARNING, 'deprecated');
294
295         if (!is_array($this->_xml))
296         {
297
298             return false;
299         }
300
301         $results = array();
302         foreach ($this->_xml as $name => $group)
303         {
304             $results[$name] = $this->getNumParams($name);
305         }
306         return $results;
307     }
308
309     /**
310      * Render all parameters.
311      *
312      * @param   string  $name   An optional name of the HTML form control. The default is 'params' if not supplied.
313      * @param   string  $group  An optional group to render.  The default group is used if not supplied.
314      *
315      * @return  array  An array of all parameters, each as array of the label, the form element and the tooltip.
316      *
317      * @deprecated  12.1
318      * @since   11.1
319      */
320     public function getParams($name = 'params', $group = '_default')
```

```
321     {
322
323         // Deprecation warning.
324         JLog::add('JParameter::getParams is deprecated.', JLog::WARNING, 'deprecated');
325
326         if (!isset($this->_xml[$group]))
327         {
328
329             return false;
330         }
331
332         $results = array();
333         foreach ($this->_xml[$group]->children() as $param)
334         {
335             $results[] = $this->getParam($param, $name, $group);
336         }
337         return $results;
338     }
339
340     /**
341      * Render a parameter type.
342      *
343      * @param   object  &$node         A parameter XML element.
344      * @param   string  $control_name  An optional name of the HTML form control. The default is 'params' if not supplied.
345      * @param   string  $group         An optional group to render.  The default group is used if not supplied.
346      *
347      * @return  array  Any array of the label, the form element and the tooltip.
348      *
349      * @deprecated  12.1
350      * @since   11.1
351      */
352     public function getParam(&$node, $control_name = 'params', $group = '_default')
353     {
354         // Deprecation warning.
355         JLog::add('JParameter::__construct is deprecated.', JLog::WARNING, 'deprecated');
356
357         // Get the type of the parameter.
358         $type = $node->attributes('type');
359
360         $element = $this->loadElement($type);
361
362         // Check for an error.
363         if ($element === false)
364         {
365             $result = array();
366             $result[0] = $node->attributes('name');
367             $result[1] = JText::_('Element not defined for type') . ' = ' . $type;
368             $result[5] = $result[0];
369             return $result;
370         }
371
372         // Get value.
373         $value = $this->get($node->attributes('name'), $node->attributes('default'), $group);
374
375         return $element->render($node, $value, $control_name);
376     }
377
378     /**
379      * Loads an XML setup file and parses it.
380      *
381      * @param   string  $path  A path to the XML setup file.
382      *
383      * @return  object
384      *
385      * @deprecated  12.1
386      * @since   11.1
387      */
388     public function loadSetupFile($path)
389     {
390         $result = false;
391
392         if ($path)
393         {
394             $xml = JFactory::getXMLParser('Simple');
395
396             if ($xml->loadFile($path))
397             {
398                 if ($params = $xml->document->params)
399                 {
400                     foreach ($params as $param)
```

```
401                 {
402                     $this->setXML($param);
403                     $result = true;
404                 }
405             }
406         }
407     }
408     else
409     {
410         $result = true;
411     }
412
413     return $result;
414 }
415
416 /**
417  * Loads an element type.
418  *
419  * @param   string   $type  The element type.
420  * @param   boolean  $new   False (default) to reuse parameter elements; true to load the parameter element type again.
421  *
422  * @return  object
423  *
424  * @deprecated  12.1
425  * @since   11.1
426  */
427 public function loadElement($type, $new = false)
428 {
429     $signature = md5($type);
430
431     if ((isset($this->_elements[$signature]) && !($this->_elements[$signature] instanceof __PHP_Incomplete_Class)) && $new === false)
432     {
433         return $this->_elements[$signature];
434     }
435
436     $elementClass = 'JElement' . $type;
437     if (!class_exists($elementClass))
438     {
439         if (isset($this->_elementPath))
440         {
441             $dirs = $this->_elementPath;
442         }
443         else
444         {
445             $dirs = array();
446         }
447
448         $file = JFilterInput::getInstance()->clean(str_replace('_', DS, $type) . '.php', 'path');
449
450         jimport('joomla.filesystem.path');
451         if ($elementFile = JPath::find($dirs, $file))
452         {
453             include_once $elementFile;
454         }
455         else
456         {
457             $false = false;
458             return $false;
459         }
460     }
461
462     if (!class_exists($elementClass))
463     {
464         $false = false;
465         return $false;
466     }
467
468     $this->_elements[$signature] = new $elementClass($this);
469
470     return $this->_elements[$signature];
471 }
472
473 /**
474  * Add a directory where JParameter should search for element types.
475  *
476  * You may either pass a string or an array of directories.
477  *
478  * JParameter will be searching for a element type in the same
479  * order you added them. If the parameter type cannot be found in
480  * the custom folders, it will look in
```

```
481     * JParameter/types.
482     *
483     * @param   mixed  $path  Directory (string) or directories (array) to search.
484     *
485     * @return  void
486     *
487     * @deprecated  12.1
488     * @since   11.1
489     */
490    public function addElementPath($path)
491    {
492        // Just force path to array.
493        settype($path, 'array');
494
495        // Loop through the path directories.
496        foreach ($path as $dir)
497        {
498            // No surrounding spaces allowed!
499            $dir = trim($dir);
500
501            // Add trailing separators as needed.
502            if (substr($dir, -1) != DIRECTORY_SEPARATOR)
503            {
504                // Directory
505                $dir .= DIRECTORY_SEPARATOR;
506            }
507
508            // Add to the top of the search dirs.
509            array_unshift($this->_elementPath, $dir);
510        }
511    }
512 }
513
```