

v1.7.16.1 ▾

...

openlitespeed / dist / admin / html.open / lib / CValidation.php / <> Jump to ▾



gwanglst Check in 1.7.16.1

History

3 contributors



878 lines (768 sloc) | 26.9 KB

...

```

1  <?php
2
3  class CValidation
4  {
5
6      protected $_disp;
7      protected $_go_flag;
8
9      public function __construct()
10     {
11     }
12
13     public function ExtractPost($disp)
14     {
15         $this->_disp = $disp;
16         $this->_go_flag = 1;
17
18         $tid = $disp->GetLast(DInfo::FLD_TID);
19         $tbl = DTblDef::GetInstance()->GetTblDef($tid);
20
21         $extracted = new CNode(CNode::K_EXTRACTED, '', CNode::T_KB);
22         $attrs = $tbl->Get(DTbl::FLD_DATTRS);
23
24         foreach ($attrs as $attr) {
25
26             if ($attr->bypassSavePost()) {
27                 continue;
28             }
29

```

```

30         $needCheck = $attr->extractPost($extracted);
31
32         if ($needCheck) {
33             if ($attr->_type == 'sel1' || $attr->_type == 'sel2') {
34                 if ($this->_disp->Get(DInfo::FLD_ACT) == 'c') {
35                     $needCheck = false; // for changed top category
36                 } else {
37                     $attr->SetDerivedSelOptions($disp->GetDerivedSelOptions($tid, $attr->_minV
38                 )
39             }
40             $dlayer = $extracted->GetChildren($attr->GetKey());
41             if ($needCheck) {
42                 $this->validateAttr($attr, $dlayer);
43             }
44             if (($tid == 'V_TOPD' || $tid == 'V_BASE') && $attr->_type == 'vhname') {
45                 $vhname = $dlayer->Get(CNode::FLD_VAL);
46                 $disp->Set(DInfo::FLD_ViewName, $vhname);
47             }
48         }
49     }
50
51     $res = $this->validatePostTbl($tbl, $extracted);
52     $this->setValid($res);
53
54     // if 0 , make it always point to curr page
55
56     if ($this->_go_flag <= 0) {
57         $extracted->SetErr('Input error detected. Please resolve the error(s). ');
58     }
59
60     $this->_disp = null;
61     return $extracted;
62 }
63
64 protected function setValid($res)
65 {
66     if ($this->_go_flag != -1) {
67         if ($res == -1) {
68             $this->_go_flag = -1;
69         } elseif ($res == 0 && $this->_go_flag == 1) {
70             $this->_go_flag = 0;
71         }
72     }
73     if ($res == 2) {
74         $this->_go_flag = 2;
75     }
76 }
77
78 protected function validatePostTbl($tbl, $extracted)

```

```

79     {
80         $tid = $tbl->Get(DTbl::FLD_ID);
81
82         if (($index = $tbl->Get(DTbl::FLD_INDEX)) != null) {
83             $keynode = $extracted->GetChildren($index);
84             if ($keynode != null) {
85                 $holderval = $keynode->Get(CNode::FLD_VAL);
86                 $extracted->SetVal($holderval);
87
88                 if ($holderval != $this->_disp->GetLast(DInfo::FLD_REF)) {
89                     // check conflict
90                     $ref = $this->_disp->GetParentRef();
91                     $location = DPageDef::GetPage($this->_disp)->GetTblMap()->FindTblLoc($tid);
92                     if ($location[0] == '*') { // allow multiple
93                         $confdata = $this->_disp->Get(DInfo::FLD_ConfData);
94                         $existingkeys = $confdata->GetChildrenValues($location, $ref);
95
96                         if (in_array($holderval, $existingkeys)) {
97                             $keynode->SetErr('This value has been used! Please choose a unique one
98                             return -1;
99                         }
100                     }
101                 }
102             }
103         }
104
105         if (($defaultExtract = $tbl->Get(DTbl::FLD_DEFAULTEXTRACT)) != null) {
106             foreach ($defaultExtract as $k => $v) {
107                 $extracted->AddChild(new CNode($k, $v));
108             }
109         }
110
111         $view = $this->_disp->Get(DInfo::FLD_View);
112         if ($tid == 'L_GENERAL' || $tid == 'ADM_L_GENERAL') {
113             return $this->chkPostTbl_L_GENERAL($extracted);
114         } elseif ($view == 'sl' || $view == 'al') { // will ignore vhlevel
115             if ($tid == 'LVT_SSL_CERT') {
116                 return $this->chkPostTbl_L_SSL_CERT($extracted);
117             }
118         }
119         elseif ($view == 'admin') {
120             if ($tid == 'ADM_USR') {
121                 return $this->chkPostTbl_ADM_USR($extracted);
122             }
123             elseif ($tid == 'ADM_USR_NEW') {
124                 return $this->chkPostTbl_ADM_USR_NEW($extracted);
125             }
126         }
127         elseif ($tid == 'V_UBND') {

```

```

128         return $this->chkPostTbl_ADM_USR_NEW($extracted);
129     }
130
131     return 1;
132 }
133
134 protected function encryptPass($val)
135 {
136     $valid_chars = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789/.";
137     $limit = strlen($valid_chars) - 1;
138     $isMac = (strtoupper(PHP_OS) === 'DARWIN');
139
140     if (CRYPT_MD5 == 1 && !$isMac) {
141         $salt = '$1$';
142         for ($i = 0; $i < 8; $i++) {
143             $salt .= $valid_chars[rand(0, $limit)];
144         }
145         $salt .= '$';
146     } else {
147         $salt = $valid_chars[rand(0, $limit)];
148         $salt .= $valid_chars[rand(0, $limit)];
149     }
150     $pass = crypt($val, $salt);
151     return $pass;
152 }
153
154 protected function chkPostTbl_ADM_USR($d)
155 {
156     $isValid = 1;
157     $oldpass = $d->GetChildVal('oldpass');
158     if ($oldpass == null) {
159         $d->SetChildErr('oldpass', 'Missing Old password!');
160         $isValid = -1;
161     } else {
162         $file = SERVER_ROOT . 'admin/conf/htpasswd';
163         $udb = $this->_disp->Get(DInfo::FLD_ConfData);
164
165         $oldusername = $this->_disp->GetLast(DInfo::FLD_REF);
166         $passwd = $udb->GetChildVal('*index$name:passwd', $oldusername);
167
168         $encrypt = crypt($oldpass, $passwd);
169
170         if ($encrypt != $passwd) {
171             $d->SetChildErr('oldpass', 'Invalid old password!');
172             $isValid = -1;
173         }
174     }
175
176     $pass = $d->GetChildVal('pass');

```

```

177         if ($pass == null) {
178             $d->SetChildErr('pass', 'Missing new password!');
179             $isValid = -1;
180         } elseif ($pass != $d->GetChildVal('pass1')) {
181             $d->SetChildErr('pass', 'New passwords do not match!');
182             $isValid = -1;
183         }
184
185         if ($isValid == -1) {
186             return -1;
187         }
188
189         $newpass = $this->encryptPass($pass);
190         $d->AddChild(new CNode('passwd', $newpass));
191         return 1;
192     }
193
194     protected function chkPostTbl_ADM_USR_NEW($d)
195     {
196         $isValid = 1;
197         $pass = $d->GetChildVal('pass');
198         if ($pass == null) {
199             $d->SetChildErr('pass', 'Missing new password!');
200             $isValid = -1;
201         } elseif ($pass != $d->GetChildVal('pass1')) {
202             $d->SetChildErr('pass', 'New passwords do not match!');
203             $isValid = -1;
204         }
205
206         if ($isValid == -1) {
207             return -1;
208         }
209
210         $newpass = $this->encryptPass($pass);
211         $d->AddChild(new CNode('passwd', $newpass));
212
213         return 1;
214     }
215
216     protected function chkPostTbl_L_GENERAL($d)
217     {
218         $isValid = 1;
219         $ip = $d->GetChildVal('ip');
220         $port = $d->GetChildVal('port');
221
222         $is_v6ip = ($ip == '[ANY]') || (strpos($ip, ':') !== false);
223
224         $confdata = $this->_disp->Get(DInfo::FLD_ConfData);
225         $lastref = $this->_disp->GetLast(DInfo::FLD_REF);

```

```

226     $nodes = $confdata->GetRootNode()->GetChildren('listener');
227
228     foreach ($nodes as $ref => $node) {
229         if ($ref == $lastref) {
230             continue;
231         }
232         $nodeport = $node->GetChildVal('port');
233         if ($port != $nodeport) {
234             continue;
235         }
236
237         $nodeip = $node->GetChildVal('ip');
238         $is_v6node = ($nodeip == '[ANY]') || (strpos($nodeip, ':') !== false);
239         if (($ip == $nodeip)
240             || ($ip == '[ANY]' && $is_v6node) || ($is_v6ip && $nodeip == '[ANY]')
241             || ($ip == 'ANY' && !$is_v6node) || (!$is_v6ip && $nodeip == 'ANY')) {
242             // ANY is IPv4, [ANY] is IPv6
243             $d->SetChildErr('port', 'This port is already in use.');
```

244 \$isValid = -1;

245 break;

246 }

247 }

248

249 \$ip0 = (\$ip == 'ANY') ? '*' : \$ip;

250 \$d->AddChild(new CNode('address', "\$ip0:\$port"));

251 return \$isValid;

252 }

253

254 protected function isCurrentListenerSecure()

255 {

256 \$confdata = \$this->_disp->Get(DInfo::FLD_ConfData);

257 \$listener = \$confdata->GetChildNodeById('listener', \$this->_disp->Get(DInfo::FLD_ViewName)

258 \$secure = \$listener->GetChildVal('secure');

259 return (\$secure == 1);

260 }

261

262 protected function chkPostTbl_L_SSL_CERT(\$d)

263 {

264 \$isValid = 1;

265 if (\$this->isCurrentListenerSecure()) {

266 \$err = 'Value must be set for secured listener. ';

267 if (\$d->GetChildVal('keyFile') == null) {

268 \$d->SetChildErr('keyFile', \$err);

269 \$isValid = -1;

270 }

271 if (\$d->GetChildVal('certFile') == null) {

272 \$d->SetChildErr('certFile', \$err);

273 \$isValid = -1;

274 }

```

275     }
276
277     return $isValid;
278 }
279
280 protected function validateAttr($attr, $dlayer)
281 {
282     if (is_array($dlayer)) {
283         foreach ($dlayer as $node) {
284             $res = $this->isValidAttr($attr, $node);
285             $this->setValid($res);
286         }
287     } else {
288         $res = $this->isValidAttr($attr, $dlayer);
289         $this->setValid($res);
290     }
291 }
292
293 protected function isValidAttr($attr, $node)
294 {
295     if ($node == null || $node->HasErr())
296         return -1;
297
298     if (!$node->HasVal()) {
299         if (!$attr->IsFlagOn(DAttr::BM_NOTNULL))
300             return 1;
301
302         $node->SetErr('value must be set. ');
303         return -1;
304     }
305
306     $notchk = array('cust', 'domain', 'subnet');
307     if (in_array($attr->_type, $notchk)) {
308         return 1;
309     }
310
311     $chktype = array('uint', 'name', 'vhname', 'dbname', 'admname', 'sel', 'sel1', 'sel2',
312                     'bool', 'file', 'filep', 'file0', 'file1', 'filetp', 'filevh', 'path', 'note',
313                     'uri', 'expuri', 'url', 'httpurl', 'email', 'dir', 'addr', 'wsaddr', 'parse');
314
315     if (!in_array($attr->_type, $chktype)) {
316         return 1;
317     }
318
319     $type3 = substr($attr->_type, 0, 3);
320     if ($type3 == 'sel') {
321         // for sel, sel1, sel2
322         $funcname = 'chkAttr_sel';
323     } elseif ($type3 == 'fil' || $type3 == 'pat') {

```

```

324         $funcname = 'chkAttr_file';
325     } else {
326         $funcname = 'chkAttr_' . $attr->_type;
327     }
328
329     if ($attr->_multiInd == 1) {
330         $vals = preg_split("/", "/", $node->Get(CNode::FLD_VAL), -1, PREG_SPLIT_NO_EMPTY);
331         $err = [];
332         $funcname .= '_val';
333         foreach ($vals as $i => $v) {
334             $res = $this->$funcname($attr, $v, $err[$i]);
335             $this->setValid($res);
336         }
337         $error = trim(implode(' ', $err));
338         if ($error != '')
339             $node->SetErr($error);
340         return 1;
341     } else {
342         return $this->$funcname($attr, $node);
343     }
344 }
345
346 protected function chkAttr_sel($attr, $node)
347 {
348     $err = '';
349     $res = $this->chkAttr_sel_val($attr, $node->Get(CNode::FLD_VAL), $err);
350     if ($err != '')
351         $node->SetErr($err);
352     return $res;
353 }
354
355 protected function chkAttr_sel_val($attr, $val, &$err)
356 {
357     if (isset($attr->_maxVal) && !array_key_exists($val, $attr->_maxVal)) {
358         $err = "invalid value: $val";
359         return -1;
360     }
361     return 1;
362 }
363
364 protected function chkAttr_admname($attr, $node)
365 {
366     $val = preg_replace("/\s+/", ' ', $node->Get(CNode::FLD_VAL));
367     $node->SetVal($val);
368     $err = '';
369     if (strlen($val) > 25) {
370         $err = 'name cannot be longer than 25 characters';
371     } else {
372         $v1 = escapeshellcmd($val);

```



```

373         if ($v1 !== $val) {
374             $err = 'invalid characters in name';
375         }
376     }
377     if ($err != '') {
378         $node->SetErr($err);
379         return -1;
380     }
381     return 1;
382 }
383
384 protected function chkAttr_name($attr, $node)
385 {
386     $node->SetVal(preg_replace("/\s+/", ' ', $node->Get(CNode::FLD_VAL)));
387     $res = $this->chkAttr_name_val($attr, $node->Get(CNode::FLD_VAL), $err);
388     if ($err != '')
389         $node->SetErr($err);
390     return $res;
391 }
392
393 protected function chkAttr_name_val($attr, $val, &$err)
394 {
395     if (preg_match("/[{}<>%]/", $val)) {
396         $err = 'invalid characters in name';
397         return -1;
398     }
399     if (strlen($val) > 100) {
400         $err = 'name cannot be longer than 100 characters';
401         return -1;
402     }
403     return 1;
404 }
405
406 protected function chkAttr_note($attr, $node)
407 {
408     if (preg_match("/[{}<]/", $node->Get(CNode::FLD_VAL), $m)) { // avoid <script, also {} for
409         $node->SetErr("character $m[0] not allowed");
410         return -1;
411     }
412     return 1;
413 }
414
415 protected function chkAttr_dbname($attr, $node)
416 {
417     return $this->chkAttr_vhname($attr, $node);
418 }
419
420 protected function chkAttr_vhname($attr, $node)
421 {

```

```

422     $node->SetVal(preg_replace("/\s+/", ' ', $node->Get(CNode::FLD_VAL)));
423     $val = $node->Get(CNode::FLD_VAL);
424     if (preg_match("/[,;<>&%=\(\)\\"'/", $val)) {
425         $node->SetErr('Invalid characters found in name');
426         return -1;
427     }
428     if (strpos($val, ' ') !== false) {
429         $node->SetErr('No space allowed in the name');
430         return -1;
431     }
432     if (strlen($val) > 100) {
433         $node->SetErr('name can not be longer than 100 characters');
434         return -1;
435     }
436     return 1;
437 }
438
439 protected function allow_create($attr, $absname)
440 {
441     if (strpos($attr->_maxVal, 'c') === false)
442         return false;
443
444     if ($attr->_minVal >= 2 && ( strpos($absname, SERVER_ROOT) === 0 )) {
445         return true;
446     }
447     //other places need to manually create
448     return false;
449 }
450
451 protected function get_cleaned_abs_path($attr_minVal, &$amp;path, &$amp;err)
452 {
453     if ($this->get_abs_path($attr_minVal, $path, $err) == 1) {
454         $absname = $this->clean_absolute_path($path);
455         return $absname;
456     }
457     return null;
458 }
459
460 protected function clean_absolute_path($abspath)
461 {
462     $absname = PathTool::clean($abspath);
463     if ( isset( $_SERVER['LS_CHROOT'] ) ) {
464         $root = $_SERVER['LS_CHROOT'];
465         $len = strlen($root);
466         if ( strncmp( $absname, $root, $len ) == 0 ) {
467             $absname = substr($absname, $len);
468         }
469     }
470     return $absname;

```

```

471     }
472
473     protected function test_file(&$absname, &$err, $attr)
474     {
475         if ($attr->_maxVal == null)
476             return 1; // no permission test
477
478         $absname = $this->clean_absolute_path($absname);
479
480         if ($attr->_type == 'file0') {
481             if (!file_exists($absname)) {
482                 return 1; //allow non-exist
483             }
484         }
485         if ($attr->_type == 'path' || $attr->_type == 'filep' || $attr->_type == 'dir') {
486             $type = 'path';
487         } else {
488             $type = 'file';
489         }
490
491         if (($type == 'path' && !is_dir($absname)) || ($type == 'file' && !is_file($absname))) {
492             $err = $type . ' ' . htmlspecialchars($absname) . ' does not exist.';
493             if ($this->allow_create($attr, $absname)) {
494                 $err .= ' <a href="javascript:lst_createFile(\'\' . $attr->GetKey() . '\')">CLICK T
495             } else {
496                 $err .= ' Please create manually.';
497             }
498
499             return -1;
500         }
501         if ((strpos($attr->_maxVal, 'r') !== false) && !is_readable($absname)) {
502             $err = $type . ' ' . htmlspecialchars($absname) . ' is not readable';
503             return -1;
504         }
505         if ((strpos($attr->_maxVal, 'w') !== false) && !is_writable($absname)) {
506             $err = $type . ' ' . htmlspecialchars($absname) . ' is not writable';
507             return -1;
508         }
509
510         return 1;
511     }
512
513     protected function chkAttr_file($attr, $node)
514     {
515         $val = $node->Get(CNode::FLD_VAL);
516         $err = '';
517         $res = $this->chkAttr_file_val($attr, $val, $err);
518         $node->SetVal($val);
519         if ($err != '')

```

```

520         $node->SetErr($err);
521     return $res;
522 }
523
524 protected function chkAttr_dir($attr, $node)
525 {
526     $val = $node->Get(CNode::FLD_VAL);
527     $err = '';
528
529     if (substr($val, -1) == '*') {
530         $res = $this->chkAttr_file_val($attr, substr($val, 0, -1), $err);
531     } else {
532         $res = $this->chkAttr_file_val($attr, $val, $err);
533     }
534     $node->SetVal($val);
535     if ($err != '')
536         $node->SetErr($err);
537     return $res;
538 }
539
540 protected function isNotAllowedPath($path)
541 {
542     $blocked = '/admin/html/';
543     if (strpos($path, $blocked) !== false) {
544         return true;
545     }
546     return false;
547 }
548
549 protected function isNotAllowedExtension($path)
550 {
551     $ext = substr($path, -4);
552     $notallowed = ['.php', '.cgi', '.pl', '.shtml'];
553     foreach ($notallowed as $test) {
554         if (strcasecmp($ext, $test) == 0) {
555             return true;
556         }
557     }
558     return false;
559 }
560
561 protected function check_cmd_invalid_str($cmd)
562 {
563     // check if it's allowed command, do not allow ' " -c -i /dev/tcp bash sh csh tcsh ksh zsh
564     $cmd = str_replace('.', 'a', $cmd); // replace . with char before pattern check
565     $pattern = '#("|\'|;|-c|-i|/dev/tcp|curl|wget|fetch|\Wbash\W|\Wsh\W|\Wcsh\W|\Wtcsh\W|\Wzsh
566
567     if (preg_match($pattern, $cmd, $m)) {
568         return $m[0];

```

```

569     }
570     $cmd = str_replace('\\', '', $cmd); // remove all escape & try again
571     if (preg_match($pattern, $cmd, $m)) {
572         return $m[0];
573     }
574     return null;
575 }
576
577 public function chkAttr_file_val($attr, $val, &$err)
578 {
579     // apply to all
580     if ($this->isNotAllowedPath($val)) {
581         $err = 'Directory not allowed';
582         return -1;
583     }
584     if ( $attr->_type == 'file0' && $this->isNotAllowedExtension($val)) {
585         $err = 'File extension not allowed';
586         return -1;
587     }
588
589     clearstatcache();
590     $err = null;
591
592     if ($attr->_type == 'filep') {
593         $path = substr($val, 0, strrpos($val, '/'));
594     } else {
595         $path = $val;
596         if ($attr->_type == 'file1') { // file1 is command
597             $invalid_str = $this->check_cmd_invalid_str($path);
598             if ($invalid_str) {
599                 $err = 'Cannot contain string ' . htmlspecialchars($invalid_str, ENT_QUOTES);
600                 return -1;
601             }
602             $pos = strpos($val, ' ');
603             if ($pos > 0) {
604                 $path = substr($val, 0, $pos); // check first part is valid path
605             }
606         }
607     }
608
609     $res = $this->chk_file1($attr, $path, $err);
610
611     if ($attr->_type == 'filetp') {
612         $pathtp = SERVER_ROOT . 'conf/templates/';
613         if (strpos($path, $pathtp) === false) {
614             $err = ' Template file must locate within $SERVER_ROOT/conf/templates/';
615             $res = -1;
616         } else if (substr($path, -5) != '.conf') {
617             $err = ' Template file name needs to be ".conf"';

```

```

618         $res = -1;
619     }
620 } elseif ($attr->_type == 'filevh') {
621     $pathvh = SERVER_ROOT . 'conf/vhosts/';
622     if (strstr($path, $pathvh) === false) {
623         $err = ' VHost config file must locate within $SERVER_ROOT/conf/vhosts/, suggested
624         $res = -1;
625     } else if (substr($path, -5) != '.conf') {
626         $err = ' VHost config file name needs to be ".conf"';
627         $res = -1;
628     }
629 }
630
631 if ($res == -1 && isset($_POST['file_create']) && $_POST['file_create'] == $attr->GetKey())
632     if (PathTool::createFile($path, $err, $attr->GetKey())) {
633         $err = "$path has been created successfully.";
634     }
635     $res = 0; // make it always point to curr page
636 }
637
638 return $res;
639 }
640
641 protected function get_abs_path($attr_minVal, &$amp;path, &$amp;err)
642 {
643     if (!strlen($path)) {
644         $err = "Invalid Path.";
645         return -1;
646     }
647
648     $s = substr($path, 0, 1);
649
650     if (strpos($path, '$VH_NAME') !== false) {
651         $path = str_replace('$VH_NAME', $this->_disp->Get(DInfo::FLD_ViewName), $path);
652     }
653
654     if ($s == '/') {
655         return 1;
656     }
657
658     if ($attr_minVal == 1) {
659         $err = 'only accept absolute path. ';
660         return -1;
661     } elseif ($attr_minVal == 2) {
662         if (strncasecmp('$SERVER_ROOT', $path, 12) == 0) {
663             $path = SERVER_ROOT . substr($path, 13);
664         } elseif ($s == '$') {
665             $err = 'only accept absolute path or path relative to $SERVER_ROOT: ' . $path;
666             return -1;

```

```

667         } else {
668             $path = SERVER_ROOT . $path; // treat as relative to SERVER_ROOT
669         }
670     } elseif ($attr_minVal == 3) {
671         if (strncasecmp('$SERVER_ROOT', $path, 12) == 0) {
672             $path = SERVER_ROOT . substr($path, 13);
673         } elseif (strncasecmp('$VH_ROOT', $path, 8) == 0) {
674             $vhroot = $this->_disp->GetVHRoot();
675             if ($vhroot == null) {
676                 $err = 'Fail to find $VH_ROOT';
677                 return -1;
678             }
679             $path = $vhroot . substr($path, 9);
680         } elseif ($s == '$') {
681             $err = 'only accept absolute path or path relative to $SERVER_ROOT or $VH_ROOT: '
682             return -1;
683         } else {
684             $path = SERVER_ROOT . $path; // treat as relative to SERVER_ROOT
685         }
686     }
687
688     return 1;
689 }
690
691 protected function chk_file1($attr, &$amp;path, &$amp;err)
692 {
693     $res = $this->get_abs_path($attr->_minVal, $path, $err);
694     if ($res == 1) {
695         return $this->test_file($path, $err, $attr);
696     }
697     return $res;
698 }
699
700 protected function chkAttr_uri($attr, $node)
701 {
702     $val = $node->Get(CNode::FLD_VAL);
703     if ($val[0] != '/') {
704         $node->SetErr('URI must start with "/"');
705         return -1;
706     }
707     return 1;
708 }
709
710 protected function chkAttr_expuri($attr, $node)
711 {
712     $val = $node->Get(CNode::FLD_VAL);
713     if (substr($val, 0, 1) == '/' || strncmp($val, 'exp:', 4) == 0) {
714         return 1;
715     } else {

```

```

716         $node->SetErr('URI must start with "/" or "exp:");
717         return -1;
718     }
719 }
720
721 protected function chkAttr_url($attr, $node)
722 {
723     $val = $node->Get(CNode::FLD_VAL);
724     if (( substr($val, 0, 1) != '/' ) && ( strcmp($val, 'http://', 7) != 0 ) && ( strcmp($va
725         $node->SetErr('URL must start with "/" or "http(s)://"');
726         return -1;
727     }
728     return 1;
729 }
730
731 protected function chkAttr_httpurl($attr, $node)
732 {
733     $val = $node->Get(CNode::FLD_VAL);
734     if (strcmp($val, 'http://', 7) != 0) {
735         $node->SetErr('Http URL must start with "http://"');
736         return -1;
737     }
738     return 1;
739 }
740
741 protected function chkAttr_email($attr, $node)
742 {
743     $err = '';
744     $res = $this->chkAttr_email_val($attr, $node->Get(CNode::FLD_VAL), $err);
745     if ($err != '')
746         $node->SetErr($err);
747     return $res;
748 }
749
750 protected function chkAttr_email_val($attr, $val, &$err)
751 {
752     if (preg_match("/^[[:alnum:]._-]+@.+/", $val)) {
753         return 1;
754     } else {
755         $err = 'invalid email format: ' . $val;
756         return -1;
757     }
758 }
759
760 protected function chkAttr_addr($attr, $node)
761 {
762     $v = $node->Get(CNode::FLD_VAL);
763     if (preg_match("/^([[[:alnum:]._-]+|\\[[[:xdigit:]]+\\]]:(\\d)+$/", $v)) {
764         return 1;

```



```

765     }
766     if ($this->isUdsAddr($v)) {
767         return 1;
768     }
769
770     $node->SetErr('invalid address: correct syntax is "IPV4|IPV6_address:port" or UDS://path o
771     return -1;
772 }
773
774 protected function isUdsAddr($v)
775 {
776     // check UDS:// unix:
777     if (preg_match('/^(UDS:\\\\|unix:)(.+)$/i', $v, $m)) {
778         $v = $m[2];
779         $supportedvar = ['$SERVER_ROOT', '$VH_NAME', '$VH_ROOT', '$DOC_ROOT'];
780         $v = str_replace($supportedvar, 'VAR', $v);
781         if (preg_match("/^[a-z0-9\\-\\_\\.]+$/i", $v)) {
782             return 1;
783         }
784     }
785     return 0;
786 }
787
788 protected function chkAttr_wsaddr($attr, $node)
789 {
790     $v = $node->Get(CNode::FLD_VAL);
791     if (preg_match("/^((http|https):\\\\|\\)?([[:alnum:]]_\\-|\\[[[:xdigit:]]+\\])(:\\d+)?$/", $v))
792         return 1;
793     }
794     if ($this->isUdsAddr($v)) {
795         return 1;
796     }
797
798     $node->SetErr('invalid address: correct syntax is "[http|https://]IPV4|IPV6_address[:port]
799     return -1;
800 }
801
802 protected function chkAttr_bool($attr, $node)
803 {
804     $val = $node->Get(CNode::FLD_VAL);
805     if ($val === '1' || $val === '0') {
806         return 1;
807     }
808     $node->SetErr('invalid value');
809     return -1;
810 }
811
812 protected function chkAttr_parse($attr, $node)
813 {

```

```

814     $err = '';
815     $res = $this->chkAttr_parse_val($attr, $node->Get(CNode::FLD_VAL), $err);
816     if ($err != '')
817         $node->SetErr($err);
818     return $res;
819 }
820
821 protected function chkAttr_parse_val($attr, $val, &$err)
822 {
823     if (preg_match($attr->_minVal, $val)) {
824         return 1;
825     } else {
826         if ($attr->_maxVal) { // has parse_help
827             $err = "invalid format \"$val\". Syntax is {$attr->_minVal} - {$attr->_maxVal}";
828         } else {
829             // when no parse_help, do not show syntax, e.g. used for not allowed value.
830             $err = "invalid value \"$val\".";
831         }
832         return -1;
833     }
834 }
835
836 protected function getKNum($strNum)
837 {
838     $tag = strtoupper(substr($strNum, -1));
839     switch ($tag) {
840         case 'K': $multi = 1024;
841             break;
842         case 'M': $multi = 1048576;
843             break;
844         case 'G': $multi = 1073741824;
845             break;
846         default: return intval($strNum);
847     }
848
849     return (intval(substr($strNum, 0, -1)) * $multi);
850 }
851
852 protected function chkAttr_uint($attr, $node)
853 {
854     $val = $node->Get(CNode::FLD_VAL);
855     if (preg_match("/^(-?\d+)([KkMmGg]?)$/", $val, $m)) {
856         $val1 = $this->getKNum($val);
857         if (isset($attr->_minVal)) {
858             $min = $this->getKNum($attr->_minVal);
859             if ($val1 < $min) {
860                 $node->SetErr('number is less than the minimum required');
861                 return -1;
862             }

```

```
863     }
864     if (isset($attr->_maxVal)) {
865         $max = $this->getKNum($attr->_maxVal);
866         if ($val1 > $max) {
867             $node->SetErr('number exceeds maximum allowed');
868             return -1;
869         }
870     }
871     return 1;
872 } else {
873     $node->SetErr('invalid number format');
874     return -1;
875 }
876 }
877
878 }
```