## Contents

1 Texis Web Script ................................................................. 1  
   1.1 Overview ................................................................. 1  
      1.1.1 Directives ....................................................... 2  
      1.1.2 Functions ...................................................... 2  
      1.1.3 Statements ..................................................... 3  
      1.1.4 Variables ....................................................... 4  
      1.1.5 Execution ....................................................... 7  
      1.1.6 URL Syntax ..................................................... 8  
   1.2 Vortex Statements ..................................................... 11  
      1.2.1 A – function definition ...................................... 12  
      1.2.2 IF, ELSE, ELSEIF – conditional execution ............... 17  
      1.2.3 SWITCH – multi-choice branch ............................... 19  
      1.2.4 Variable assignment – assign values to variables ....... 21  
      1.2.5 LOCAL – declare local variables ............................ 23  
      1.2.6 LOOP – iterate through variables ........................... 25  
      1.2.7 SQL – execute SQL statement ............................... 27  
      1.2.8 DB – set database path ...................................... 37  
      1.2.9 USER, PASS – set SQL user and password ................. 38  
      1.2.10 WHILE – conditional loop .................................... 39  
      1.2.11 READLN – read file a line at a time ...................... 40  
      1.2.12 WRITE – write to file ....................................... 42  
      1.2.13 EXEC – execute program ..................................... 44  
      1.2.14 CAPTURE – capture output ................................... 49
1.2.15 TIMPORT – general purpose data import ........................................ 51
1.2.16 BREAK – exit loop .............................................................. 53
1.2.17 CONTINUE – continue loop .................................................... 54
1.2.18 RETURN – exit current function .............................................. 55
1.2.19 VERB – raw output ............................................................... 56

1.3 Vortex Directives ................................................................. 58
1.3.1 TIMEOUT – set timeout ........................................................ 59
1.3.2 EXPORT – retain variables across invocations ............................. 61
1.3.3 PUTMSG – set error message actions ....................................... 65
1.3.4 ENTRYFUNC, EXITFUNC – set entry/exit functions .................. 67
1.3.5 USES – use a module ............................................................ 69
1.3.6 SCHEDULE – schedule periodic execution of script ..................... 71
1.3.7 COOKIES – control cookie import ........................................ 74
1.3.8 STACK – set stack limit ........................................................ 75
1.3.9 TRACESQL – trace SQL calls ................................................ 76
1.3.10 SQLCACHE – control SQL handle caching ............................... 78
1.3.11 TRAP – trap signals ............................................................ 79
1.3.12 ADDTRAILINGSLASH – add trailing slash to “directory” URLs .... 81
1.3.13 pragma – inline compiler directive ........................................ 82

1.4 Special Variables ................................................................. 84
1.4.1 $cmd ................................................................. 84
1.4.2 $cmdlnargs ............................................................. 84
1.4.3 $errnum, $errscript, $errline etc. ........................................ 84
1.4.4 $indexcount ............................................................. 84
1.4.5 $rows.min, $rows.max .................................................. 84
1.4.6 $sqlresult.... vars .................................................. 85
1.4.7 $loop ................................................................. 85
1.4.8 $next ................................................................. 85
1.4.9 $null ................................................................. 85
1.4.10 $pathroot ............................................................ 85
CONTENTS

1.4.11 $ret ........................................ 86
1.4.12 $ret.count ................................ 86
1.4.13 $ret.seq ................................ 86
1.4.14 $ret.off ................................ 86
1.4.15 $ret.err,$ret.owner,$ret.group,$ret.size,etc. ....... 86
1.4.16 $sourcepath ................................ 86
1.4.17 $url ...................................... 87
1.4.18 $urlfunc,$urlext .......................... 87
1.4.19 $urlq ..................................... 87
1.4.20 $urllroot .................................. 87
1.4.21 $userpath .................................. 87
1.4.22 $FILE ...................................... 87
1.4.23 $LINE ..................................... 88
1.4.24 $SCRIPT ................................... 88
1.4.25 $FUNCTION ................................ 88

1.5 Builtin Functions .................................................. 89
1.5.1 sum – return sum of variable values ...................... 90
1.5.2 fmt, strfmt – formatted output .......................... 92
1.5.3 fmtcp – set fmt control parameters ..................... 107
1.5.4 fmtinfo – get fmt control parameters .................. 110
1.5.5 mm – enable/disable automatic hit markup .......... 111
1.5.6 sb – enable/disable automatic search and replace .... 113
1.5.7 apicp – modify Metamorph query control parameters . 114
1.5.8 apiinfo – get current Metamorph query control parameters 125
1.5.9 sqlcp – modify low-level SQL control parameters .... 126
1.5.10 pagelinks – paginate SQL results ....................... 133
1.5.11 read – read files ................................ 139
1.5.12 send – print raw data ........................... 140
1.5.13 spew – print files ................................ 141
1.5.14 flush – flush output ............................ 142
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>header</td>
<td>print HTTP header</td>
<td>144</td>
</tr>
<tr>
<td>rex, split</td>
<td>regular expression search</td>
<td>148</td>
</tr>
<tr>
<td>sandr</td>
<td>regular expression search and replace</td>
<td>155</td>
</tr>
<tr>
<td>strstr, strstri</td>
<td>find substring in string</td>
<td>157</td>
</tr>
<tr>
<td>substr</td>
<td>extract substring from string</td>
<td>159</td>
</tr>
<tr>
<td>strcmp, strcmpi</td>
<td>compare strings</td>
<td>160</td>
</tr>
<tr>
<td>strncmp, strnicmp</td>
<td>compare strings, fixed length</td>
<td>161</td>
</tr>
<tr>
<td>strlen</td>
<td>length of string</td>
<td>162</td>
</tr>
<tr>
<td>strrev</td>
<td>reverse string</td>
<td>163</td>
</tr>
<tr>
<td>upper</td>
<td>convert to upper case</td>
<td>164</td>
</tr>
<tr>
<td>lower</td>
<td>convert to lower case</td>
<td>165</td>
</tr>
<tr>
<td>strfold</td>
<td>fold string by case</td>
<td>166</td>
</tr>
<tr>
<td>strfoldcmp</td>
<td>compare strings folded by case</td>
<td>167</td>
</tr>
<tr>
<td>sort</td>
<td>sort variables</td>
<td>168</td>
</tr>
<tr>
<td>uniq, uniqcount</td>
<td>produce unique list of values</td>
<td>171</td>
</tr>
<tr>
<td>count</td>
<td>return number of variable values</td>
<td>173</td>
</tr>
<tr>
<td>strtonum</td>
<td>convert string to number</td>
<td>174</td>
</tr>
<tr>
<td>rand</td>
<td>generate pseudo-random number</td>
<td>175</td>
</tr>
<tr>
<td>randpick</td>
<td>return pseudo-random value of a variable</td>
<td>176</td>
</tr>
<tr>
<td>srand</td>
<td>seed the pseudo-random number generator</td>
<td>177</td>
</tr>
<tr>
<td>exit</td>
<td>exit program</td>
<td>178</td>
</tr>
<tr>
<td>fetch</td>
<td>fetch URLs</td>
<td>179</td>
</tr>
<tr>
<td>submit</td>
<td>submit HTML form</td>
<td>182</td>
</tr>
<tr>
<td>urlinfo</td>
<td>get detailed page info</td>
<td>185</td>
</tr>
<tr>
<td>urllinks, urltext</td>
<td>get formatted text, URL links</td>
<td>197</td>
</tr>
<tr>
<td>urlcp</td>
<td>modify URL control parameters</td>
<td>198</td>
</tr>
<tr>
<td>urlutil</td>
<td>URL/network utility</td>
<td>237</td>
</tr>
<tr>
<td>nslookup</td>
<td>domain name and IP address lookup</td>
<td>240</td>
</tr>
<tr>
<td>nsinfo</td>
<td>get info from last name lookup</td>
<td>242</td>
</tr>
<tr>
<td>options</td>
<td>print options list</td>
<td>244</td>
</tr>
</tbody>
</table>
CONTENTS

1.5.45  radiobutton – print radio buttons list ........................................... 246
1.5.46  checkbox – print checkbox list .............................................. 248
1.5.47  doctype – print DOCTYPE declaration .......................................... 250
1.5.48  cal – print a calendar with links .............................................. 251
1.5.49  calrule – translate a calendar rule and iterate dates ................. 255
1.5.50  calendar – print calendar form ................................................ 257
1.5.51  caldate – create date list from calendar submission ................. 259
1.5.52  clist – create comma-separated list ........................................... 260
1.5.53  slist – create Metamorph set list ............................................ 261
1.5.54  wordlist, wordcount, wordoccurrencecounts – get words and frequencies from index .......................... 262
1.5.55  createdb – create database ..................................................... 264
1.5.56  adminsql – execute arbitrary SQL ........................................... 265
1.5.57  loguser – log user accesses ..................................................... 267
1.5.58  userstats – get user statistics ................................................ 268
1.5.59  resetstats – reset user statistics ............................................. 269
1.5.60  abstract – generate an abstract from text .................................. 270
1.5.61  rmcommon – remove common prefix/suffix from text ..................... 272
1.5.62  pwencrypt – encrypt Unix-style password .................................. 273
1.5.63  encrypt – encrypt data .......................................................... 274
1.5.64  decrypt – decrypt data .......................................................... 275
1.5.65  readvars – read URL-encoded variables ..................................... 276
1.5.66  varinfo – get miscellaneous variable information ...................... 278
1.5.67  getvar – get variable values .................................................... 281
1.5.68  setvar – set a named variable .................................................. 283
1.5.69  vxcp – set Vortex control parameters ................................. 284
1.5.70  vxinfo – get miscellaneous Vortex information ......................... 292
1.5.71  hash – produce a hash or checksum for data .............................. 297
1.5.72  geo2code – latitude/longitude encoding for regional search ........ 302
1.5.73  code2geo – decode latitude/longitude encoding ....................... 304
1.5.74  pdfxml – convert Metamorph hit to PDF markup information .......... 305
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5.75</td>
<td><code>xtree</code> — maintain sorted list of values</td>
<td>308</td>
</tr>
<tr>
<td>1.5.76</td>
<td><code>profiler</code> — assist in searching query profiles</td>
<td>313</td>
</tr>
<tr>
<td>1.5.77</td>
<td><code>sleep</code> — sleep for a while</td>
<td>315</td>
</tr>
<tr>
<td>1.5.78</td>
<td><code>sysinfo</code> — get system-specific information</td>
<td>316</td>
</tr>
<tr>
<td>1.5.79</td>
<td><code>syscp</code> — set system-specific information</td>
<td>321</td>
</tr>
<tr>
<td>1.5.80</td>
<td><code>sysutil</code> — file and system utilities</td>
<td>323</td>
</tr>
<tr>
<td>1.5.81</td>
<td><code>stat</code> — obtain file size and attributes</td>
<td>327</td>
</tr>
<tr>
<td>1.5.82</td>
<td><code>getpid</code> — get process id</td>
<td>332</td>
</tr>
<tr>
<td>1.5.83</td>
<td><code>procexists</code> — see if process exists</td>
<td>333</td>
</tr>
<tr>
<td>1.5.84</td>
<td><code>kill</code> — terminate or send a signal to a process</td>
<td>334</td>
</tr>
<tr>
<td>1.5.85</td>
<td><code>loadavg</code> — return system load averages</td>
<td>335</td>
</tr>
<tr>
<td>1.6</td>
<td>XML API</td>
<td>336</td>
</tr>
<tr>
<td>1.6.1</td>
<td>Overview</td>
<td>336</td>
</tr>
<tr>
<td>1.6.2</td>
<td>Character Encodings</td>
<td>336</td>
</tr>
<tr>
<td>1.6.3</td>
<td>Tree vs. Streaming APIs</td>
<td>337</td>
</tr>
<tr>
<td>1.6.4</td>
<td>Data Types</td>
<td>339</td>
</tr>
<tr>
<td>1.6.5</td>
<td>Using the <code>xmlTree</code></td>
<td>340</td>
</tr>
<tr>
<td>1.6.6</td>
<td><code>xmlTree FAQ</code></td>
<td>349</td>
</tr>
<tr>
<td>1.6.7</td>
<td>Using the <code>xmlReader</code></td>
<td>354</td>
</tr>
<tr>
<td>1.6.8</td>
<td>Using the <code>xmlWriter</code></td>
<td>356</td>
</tr>
<tr>
<td>1.6.9</td>
<td><code>xmlReaderGetAllContent</code></td>
<td>359</td>
</tr>
<tr>
<td>1.6.10</td>
<td><code>xmlReaderGetAttribute</code></td>
<td>361</td>
</tr>
<tr>
<td>1.6.11</td>
<td><code>xmlReaderGetAttributeCount</code></td>
<td>363</td>
</tr>
<tr>
<td>1.6.12</td>
<td><code>xmlReaderGetAttributeNumber</code></td>
<td>364</td>
</tr>
<tr>
<td>1.6.13</td>
<td><code>xmlReadergetBytesConsumed</code></td>
<td>365</td>
</tr>
<tr>
<td>1.6.14</td>
<td><code>xmlReaderGetColumn</code></td>
<td>366</td>
</tr>
<tr>
<td>1.6.15</td>
<td><code>xmlReaderGetContent</code></td>
<td>367</td>
</tr>
<tr>
<td>1.6.16</td>
<td><code>xmlReaderGetDepth</code></td>
<td>368</td>
</tr>
<tr>
<td>1.6.17</td>
<td><code>xmlReaderGetEncoding</code></td>
<td>369</td>
</tr>
<tr>
<td>1.6.18</td>
<td><code>xmlReaderGetLang</code></td>
<td>370</td>
</tr>
<tr>
<td>1.6.19</td>
<td>xmlReaderGetLine</td>
<td>371</td>
</tr>
<tr>
<td>1.6.20</td>
<td>xmlReaderGetLocalName</td>
<td>372</td>
</tr>
<tr>
<td>1.6.21</td>
<td>xmlReaderGetName</td>
<td>373</td>
</tr>
<tr>
<td>1.6.22</td>
<td>xmlReaderGetNsURI</td>
<td>374</td>
</tr>
<tr>
<td>1.6.23</td>
<td>xmlReaderGetNsPrefix</td>
<td>375</td>
</tr>
<tr>
<td>1.6.24</td>
<td>xmlReaderGetType</td>
<td>376</td>
</tr>
<tr>
<td>1.6.25</td>
<td>xmlReaderGetVersion</td>
<td>378</td>
</tr>
<tr>
<td>1.6.26</td>
<td>xmlReaderIsEmptyElement</td>
<td>379</td>
</tr>
<tr>
<td>1.6.27</td>
<td>xmlReaderMoveToAttribute</td>
<td>380</td>
</tr>
<tr>
<td>1.6.28</td>
<td>xmlReaderMoveToAttributeNumber</td>
<td>381</td>
</tr>
<tr>
<td>1.6.29</td>
<td>xmlReaderMoveToElement</td>
<td>383</td>
</tr>
<tr>
<td>1.6.30</td>
<td>xmlReaderMoveToFirstAttribute</td>
<td>384</td>
</tr>
<tr>
<td>1.6.31</td>
<td>xmlReaderMoveToNextAttribute</td>
<td>386</td>
</tr>
<tr>
<td>1.6.32</td>
<td>xmlReaderNewFromFile</td>
<td>387</td>
</tr>
<tr>
<td>1.6.33</td>
<td>xmlReaderNewFromString</td>
<td>389</td>
</tr>
<tr>
<td>1.6.34</td>
<td>xmlReaderRead</td>
<td>391</td>
</tr>
<tr>
<td>1.6.35</td>
<td>xmlTreeAddChild</td>
<td>392</td>
</tr>
<tr>
<td>1.6.36</td>
<td>xmlTreeAddChildList</td>
<td>394</td>
</tr>
<tr>
<td>1.6.37</td>
<td>xmlTreeAddContent</td>
<td>396</td>
</tr>
<tr>
<td>1.6.38</td>
<td>xmlTreeAddNextSibling</td>
<td>397</td>
</tr>
<tr>
<td>1.6.39</td>
<td>xmlTreeAddPrevSibling</td>
<td>399</td>
</tr>
<tr>
<td>1.6.40</td>
<td>xmlTreeAddSibling</td>
<td>401</td>
</tr>
<tr>
<td>1.6.41</td>
<td>xmlTreeCleanup</td>
<td>403</td>
</tr>
<tr>
<td>1.6.42</td>
<td>xmlTreeClearNs</td>
<td>404</td>
</tr>
<tr>
<td>1.6.43</td>
<td>xmlTreeCopyAttribute</td>
<td>406</td>
</tr>
<tr>
<td>1.6.44</td>
<td>xmlTreeCopyAttributeList</td>
<td>407</td>
</tr>
<tr>
<td>1.6.45</td>
<td>xmlTreeCopyDoc</td>
<td>408</td>
</tr>
<tr>
<td>1.6.46</td>
<td>xmlTreeCopyNode</td>
<td>409</td>
</tr>
<tr>
<td>1.6.47</td>
<td>xmlTreeCopyNodeList</td>
<td>410</td>
</tr>
<tr>
<td>1.6.48</td>
<td>xmlTreeDumpNode</td>
<td>411</td>
</tr>
</tbody>
</table>
1.6.79 xmlTreeNewAttribute ....................................... 450
1.6.80 xmlTreeNewCDATA ......................................... 451
1.6.81 xmlTreeNewComment ......................................... 452
1.6.82 xmlTreeNewDoc ............................................... 453
1.6.83 xmlTreeNewDocFromFile ...................................... 454
1.6.84 xmlTreeNewDocFromString .................................... 456
1.6.85 xmlTreeNewElement .......................................... 458
1.6.86 xmlTreeNewNs ................................................. 459
1.6.87 xmlTreeNewPI ............................................... 461
1.6.88 xmlTreeNewText .............................................. 462
1.6.89 xmlTreeNewXPath ............................................ 463
1.6.90 xmlTreePrintDoc ............................................. 464
1.6.91 xmlTreeQuickXPath .......................................... 466
1.6.92 xmlTreeSaveDoc ............................................. 468
1.6.93 xmlTreeSetContent .......................................... 469
1.6.94 xmlTreeSetName ............................................. 470
1.6.95 xmlTreeSetNs ............................................... 471
1.6.96 xmlTreeSetNsPrefix ......................................... 473
1.6.97 xmlTreeSetNsURI ............................................. 474
1.6.98 xmlTreeSetRootElement ...................................... 475
1.6.99 xmlTreeUnlinkNode .......................................... 476
1.6.100 xmlTreeXPathExecute ...................................... 477
1.6.101 xmlTreeXPathRegisterNs ................................... 479
1.6.102 xmlTreeXPathSetContext ................................... 480
1.6.103 xmlWriterEndAttribute .................................... 482
1.6.104 xmlWriterEndCDATA ....................................... 483
1.6.105 xmlWriterEndComment ..................................... 484
1.6.106 xmlWriterEndDocument .................................... 485
1.6.107 xmlWriterEndElement ...................................... 486
1.6.108 xmlWriterEndPI ............................................ 487
1.6.109 xmlWriterGetContent ........................................... 488
1.6.110 xmlWriterNewToFile ........................................... 489
1.6.111 xmlWriterNewToString ......................................... 490
1.6.112 xmlWriterSetIndent ............................................ 491
1.6.113 xmlWriterStartAttribute ...................................... 492
1.6.114 xmlWriterStartCDATA ......................................... 493
1.6.115 xmlWriterStartComment ........................................ 494
1.6.116 xmlWriterStartDocument ...................................... 495
1.6.117 xmlWriterStartElement ....................................... 496
1.6.118 xmlWriterStartPI .............................................. 497
1.6.119 xmlWriterWrite ................................................ 498
1.6.120 xmlWriterWriteAttribute ..................................... 499
1.6.121 xmlWriterWriteCDATA ......................................... 500
1.6.122 xmlWriterWriteComment ....................................... 501
1.6.123 xmlWriterWriteElement ....................................... 502
1.6.124 xmlWriterWritePI .............................................. 503
1.6.125 xmlWriterWriteRaw ............................................ 504
1.6.126 xsltApplyStylesheet .......................................... 505
1.6.127 xsltParseStylesheetDoc ....................................... 506
1.6.128 xsltParseStylesheetFile ...................................... 507
1.6.129 xsltParseStylesheetString ................................... 508

1.7 MIME API .......................................................... 509
1.7.1 Overview ......................................................... 509
1.7.2 Data Types ....................................................... 509
1.7.3 mimeEntityGetBody .............................................. 513
1.7.4 mimeEntityGetChildNumber .................................... 515
1.7.5 mimeEntityGetContentLocation ................................. 516
1.7.6 mimeEntityGetDepth ............................................. 517
1.7.7 mimeEntityGetHeaderNames .................................... 518
1.7.8 mimeEntityGetHeaderParameterNames ......................... 519
| 1.7.9 | mimeEntityGetHeaderParameterValues | 520 |
| 1.7.10 | mimeEntityGetHeaderValue | 522 |
| 1.7.11 | mimeEntityGetImapSectionSpecification | 523 |
| 1.7.12 | mimeEntityGetMessageFilename | 524 |
| 1.7.13 | mimeEntityGetRawBody | 526 |
| 1.7.14 | mimeEntityGetRawBodyOffset | 527 |
| 1.7.15 | mimeEntityGetRawBodySize | 528 |
| 1.7.16 | mimeEntityGetRawHeaderSection | 529 |
| 1.7.17 | mimeEntityGetRawHeaderSectionOffset | 530 |
| 1.7.18 | mimeEntityGetRawHeaderSectionSize | 531 |
| 1.7.19 | mimeEntityGetRawHeaderValue | 532 |
| 1.7.20 | mimeEntityGetSafeFilename | 533 |
| 1.7.21 | mimeEntityGetSequenceNumber | 534 |
| 1.7.22 | mimeEntityGetText | 535 |
| 1.7.23 | mimeEntityGetTextCharset | 536 |
| 1.7.24 | mimeEntityGetTextFormatter | 537 |
| 1.7.25 | mimeEntityIsLastChild | 538 |
| 1.7.26 | mimeEntityIsReparented | 539 |
| 1.7.27 | mimeEntityIsStartBodyPart | 540 |
| 1.7.28 | mimeReaderGetEntity | 541 |
| 1.7.29 | mimeReaderGetFullEntity | 542 |
| 1.7.30 | mimeReaderMoveToNextEntity | 543 |
| 1.7.31 | mimeReaderMoveToNextEntitySibling | 544 |
| 1.7.32 | mimeReaderOpenEntity | 546 |
| 1.7.33 | mimeReaderOpenFile | 547 |
| 1.7.34 | mimeReaderOpenString | 548 |
| 1.7.35 | headerDecode | 549 |
| 1.7.36 | headerGetItems | 550 |
| 1.7.37 | headerItemGetParameterNames | 551 |
| 1.7.38 | headerItemGetParameterValues | 552 |
CONTENTS

1.7.39 headerMailboxGetAddress ................................. 554
1.7.40 headerMailboxGetDisplayName ............................. 555
1.7.41 mailParseAliases ........................................ 556

1.8 Vortex Library Modules ........................................ 558
  1.8.1 Vortex Module Creation ................................. 558
  1.8.2 Vortex Module Revision ................................. 559
  1.8.3 Using Modules in Scripts ................................ 559
  1.8.4 Vortex Library Option Summary .......................... 560

1.9 Vortex Scheduling .............................................. 562
  1.9.1 Listing Scheduled Scripts .............................. 562
  1.9.2 Unscheduling Scripts ................................. 563
  1.9.3 Schedule Option Summary ............................... 563

1.10 Creating User Functions ..................................... 564
  1.10.1 Declaration and Argument Syntax ....................... 564
  1.10.2 I/O .................................................. 565
  1.10.3 Compiling and Linking ................................ 565

1.11 Vortex Command Line Options ................................ 567
  1.11.1 Vortex Global Command-line Options ................... 567
  1.11.2 Vortex Action Command-line Options ................... 568
  1.11.3 Vortex Optional Command-line Options ................. 569
  1.11.4 Vortex Command-line SQL Options ...................... 571
  1.11.5 Vortex Miscellaneous Command-line Options .......... 572
  1.11.6 Differences Between Vortex, tsql and Metamorph API .. 573

1.12 Vortex texis.ini Configuration Settings ..................... 575

1.13 Vortex Error Messages ....................................... 581

2 Texis Web Server ............................................... 587
  2.1 vhtpd Overview ............................................ 587
  2.2 vhtpd Config File .......................................... 587
    2.2.1 vhtpd Log and Path Settings .......................... 588
    2.2.2 vhtpd URL Settings ................................... 590
C.7 libxml2 .......................................................... 617
C.8 Libxslt .......................................................... 617
C.9 Libexslt .......................................................... 618
C.10 ppt2html, msg2html ............................................ 618
C.11 SSL/HTTPS plugin ............................................ 618
C.12 unrar ............................................................ 621
C.13 unzip ............................................................ 622
C.14 zlib ............................................................. 623
C.15 SpiderMonkey (JavaScript-C) Engine ...................... 624
C.16 PDF/anytotx plugin ......................................... 624
C.17 thttpd - throttling HTTP server ......................... 624
C.18 prngd .......................................................... 625
C.19 GNU General Public License ............................... 625
C.20 GNU Lesser General Public License ....................... 632
C.21 GNU Library General Public License ..................... 642
C.22 Netscape Public License .................................... 651
C.23 UnixUtils ..................................................... 659
C.24 PuTTY .......................................................... 660
C.25 MIT Kerberos .................................................. 661
C.26 Cyrus SASL ..................................................... 685

D  Texis V e rsion 6 F e atures and Changes 687
D.1 Overview ......................................................... 687
   D.1.1 Query Handling .............................................. 687
   D.1.2 Search Results Improvements .............................. 688
   D.1.3 URL Fetching and Web Server Changes .................. 688
   D.1.4 Filename Changes ........................................... 689
   D.1.5 Programming Features ...................................... 689
D.2 Filename Changes ............................................... 690
   D.2.1 texis.cnf Moved - Caveat ................................. 690
   D.2.2 Default Vortex Source File Extension .vs ............... 690
D.2.3 ErrorScript, ErrorFile Initialized Differently - Caveat ............... 690
D.3 SQL and apicp Setting Changes ............................................ 691
D.3.1 Unicode textsearchmode,stringcomparemode - Caveat ............... 691
D.3.2 tsq1 apicp Defaults Same as Vortex ................................. 692
D.3.3 apicp Defaults Settable in texis.ini ................................. 692
D.3.4 texisdefaults Deprecated - Caveat .................................... 692
D.3.5 apicp Setting querysettings Added .................................. 692
D.3.6 withinmode Defaults to word span ................................. 693
D.3.7 wildsingle On by Default ........................................... 693
D.4 Programming Features ..................................................... 693
D.4.1 Multi-Part Abstracts .................................................... 693
D.4.2 Hit Highlighting Enhanced ............................................. 693
D.4.3 Multi-value SQL Parameters and str1st Changes, arrayconvert - Caveat ............................... 694
D.4.4 <sql okvars> Option .................................................... 694
D.4.5 sqlresult... Variables Returned from <sql> ....................... 694
D.4.6 Debug Syntax for Vortex Variables ................................. 694
D.4.7 <exec> Double-quotes Args with Space ........................... 694
D.4.8 XML, XSLT API Added ................................................ 695
D.4.9 int64,uint64 Types Added ........................................... 695
D.4.10 varint,int (N) Support ................................................ 695
D.4.11 Post-processing putmsg Changes - Caveat ......................... 695
D.4.12 More XHTML Compliance ............................................ 695
D.4.13 indexaccess Columns Added ....................................... 696
D.4.14 REX Set-Subtract Operator ....................................... 696
D.4.15 Vortex Scheduler Port Changed .................................... 696
D.4.16 More anytotx Headers and Formats ............................... 696
D.5 New URL Fetching Settings ................................................. 696
D.5.1 urlcp secure ............................................................ 696
D.5.2 urlcp putmsg ........................................................... 697
D.5.3 urlcp fetchmeter ....................................................... 697
## CONTENTS

D.5.4  `urlcp httpversion` .......................... 697  
D.5.5  `urlcp ftprelativepaths` .......................... 697  
D.5.6  `urlcp sslprotocols` .......................... 697  
D.5.7  `urlcp ignoreanchorframes` .......................... 697  
D.5.8  `urlcp charsetconfigfromfile` .......................... 697  

D.6  Web Server Changes .......................... 698  
D.6.1  `vhttpd AllowExt, ExcludeExt Defaults Changed` - **Caveat** .......................... 698  
D.6.2  Variables Supported in `vhttpd Settings` .......................... 698  
D.6.3  Monitor Web Server SSL Support .......................... 698  

D.7  Webinator Changes .......................... 698  
D.7.1  Meta Search .......................... 698  
D.7.2  Maintenance Section .......................... 699  
D.7.3  License Updates Applicable via Web Interface .......................... 699  
D.7.4  XML Results .......................... 699  
D.7.5  SOAP API .......................... 699  
D.7.6  Compressed HTML Support .......................... 699  
D.7.7  Faster Category Searches .......................... 699  
D.7.8  Customizable Thesaurus .......................... 699  
D.7.9  Results Authorization .......................... 700  
D.7.10  Group by Site .......................... 700  
D.7.11  Backup and Restore Settings .......................... 700  
D.7.12  Apply/Revert Appearance Changed to Buttons .......................... 700  
D.7.13  CSS Admin GUI .......................... 700  
D.7.14  Text Search Mode .......................... 700  
D.7.15  Language Analysis Module .......................... 700  
D.7.16  Extensions Renamed and Default Changed .......................... 700  
D.7.17  Exclusions Default Changed .......................... 701  

E  **Texis Version 7 Features and Changes** .......................... 703  
E.1  Webinator Features - **Caveat** .......................... 703  
E.2  Vortex Features .......................... 703
<table>
<thead>
<tr>
<th>E.2.1</th>
<th>Script Tracing</th>
<th>703</th>
</tr>
</thead>
<tbody>
<tr>
<td>E.2.2</td>
<td>REX Does Not Return Redundant Empty Hits - <strong>Caveat</strong></td>
<td>704</td>
</tr>
<tr>
<td>E.2.3</td>
<td><code>sandr</code> Infinite Loops Prevented, Matches Like <code>rex</code> - <strong>Caveat</strong></td>
<td>704</td>
</tr>
<tr>
<td>E.2.4</td>
<td><code>sum</code> Action Determined by Format - <strong>Caveat</strong></td>
<td>704</td>
</tr>
<tr>
<td>E.2.5</td>
<td><code>fmt $1/%1:</code> Codes Changed - <strong>Caveat</strong></td>
<td>704</td>
</tr>
<tr>
<td>E.2.6</td>
<td>SQL <code>output=xml</code> Does Not Consume Entities - <strong>Caveat</strong></td>
<td>705</td>
</tr>
<tr>
<td>E.2.7</td>
<td>Empty-element Statement Tags Allowed - <strong>Caveat</strong></td>
<td>705</td>
</tr>
<tr>
<td>E.2.8</td>
<td>Pragma Stacks - <strong>Caveat</strong></td>
<td>705</td>
</tr>
<tr>
<td>E.2.9</td>
<td><code>write output,flags,skiponfail</code> Options - <strong>Caveat</strong></td>
<td>705</td>
</tr>
<tr>
<td>E.2.10</td>
<td><code>getvar, setvar</code> Error Messages</td>
<td>706</td>
</tr>
<tr>
<td>E.2.11</td>
<td><code>vxcp timeouttext</code> and <code>exceptiononly</code> Added</td>
<td>706</td>
</tr>
<tr>
<td>E.2.12</td>
<td><code>vxcp</code> <code>compatibilityversion</code> Added</td>
<td>706</td>
</tr>
<tr>
<td>E.2.13</td>
<td>Cookie Acceptance Relaxed - <strong>Caveat</strong></td>
<td>706</td>
</tr>
<tr>
<td>E.2.14</td>
<td><code>xmlTreeXPathSetContext()</code> Added to XML API</td>
<td>706</td>
</tr>
<tr>
<td>E.2.15</td>
<td><code>$ret.size</code> is <code>int64</code> On All Platforms</td>
<td>706</td>
</tr>
<tr>
<td>E.3</td>
<td>SQL Features</td>
<td>707</td>
</tr>
<tr>
<td>E.3.1</td>
<td><code>strlst/IN</code> Changes and New <code>SUBSET/INTERSECT</code> Operators</td>
<td>707</td>
</tr>
<tr>
<td>E.3.2</td>
<td><code>ALTER INDEX</code> Syntax Added</td>
<td>708</td>
</tr>
<tr>
<td>E.3.3</td>
<td><code>CREATE INDEX ... WITH options</code> Syntax - <strong>Caveat</strong></td>
<td>709</td>
</tr>
<tr>
<td>E.3.4</td>
<td>Metamorph Index Default Type Is <code>INVERTED</code> - <strong>Caveat</strong></td>
<td>709</td>
</tr>
<tr>
<td>E.3.5</td>
<td><code>byte/char</code> Conversion As-is Without Hexifying - <strong>Caveat</strong></td>
<td>709</td>
</tr>
</tbody>
</table>
Chapter 1

Texis Web Script

This document describes the Texis Web Script (Vortex) language and associated utilities. The information contained within may be updated or changed in future versions.

1.1 Overview

Texis Web Script is a powerful HTML server-side programming environment that enables a designer to easily create, deploy, and maintain Web based applications. It tightly integrates into one package three of the most needed entities:

- An object enabled SQL RDBMS (Texis)
- Concept Based Text Retrieval (Metamorph)
- A CGI-scripting language and compiler (Vortex)

The fundamental idea behind Texis Web Script is to extend HTML with programming capabilities which are directed towards the most commonly performed Web activities. This extended HTML language is called Vortex.

Vortex HTML is compiled into a P-code similar to Java. Execution of this P-code is very fast and efficient in contrast to interpreted script languages, and can easily support the demands of very highly hit Web servers.

Technical Details

Vortex scripts reside in HTML-like text files on a Web server, typically in the /usr/local/morph3/texis/scripts (Unix) or c:\morph3\texis\scripts (Windows) directory (see ScriptRoot, p. 576). The source files typically have a .vs extension, and the object files have a .vtx extension. Vortex scripts are run by the CGI program texis upon request of a Web browser (or command line). The program will automatically re-compile scripts whose objects are out of date with respect to their source, so no explicit re-compiling by the user is needed.
A Vortex script is delimited in an HTML file by `<SCRIPT>` tags, with `LANGUAGE=vortex`:

```html
...<SCRIPT LANGUAGE=vortex>
  ... Vortex directives (if any) ...
  ... Vortex functions ...
</SCRIPT>
... other HTML (ignored) ...
```

There can be multiple `<SCRIPT>` elements in a file; they are concatenated as one Vortex script (though functions cannot be split across script blocks). All text outside of `<SCRIPT language=vortex>` tags is ignored.

The syntax of Vortex scripts, with few exceptions, follows that of HTML. This allows the easy intermixing of the two, and minimizes errors if a Web browser ever views a Vortex source file or script. All Vortex commands and directives are HTML tags, some with optional attributes. Like HTML, attribute values may contain whitespace and other non-alphanumeric characters if the value is enclosed in single or double quotes. Tags and attributes are case-insensitive; however function and variable names are case-sensitive.

The following is a minimal example script that just prints “Hello, world!”:

```html
<SCRIPT LANGUAGE=vortex>

  <A NAME=main>
    Hello, world!
  </A>

</SCRIPT>
```

### 1.1.1 Directives

Vortex *directives* are commands that affect the behavior of the script, but have a global effect and are evaluated only at compile time. As such, they must appear before the first function in the script. For example, the `EXPORT` directive (p. 61) flags which variables will be exported; the actual export of these variables, however, happens when `$url` is printed at run time.

### 1.1.2 Functions

Functions execute statements or print HTML. There are 3 kinds of Vortex functions: script functions written in the script; user functions, linked into Vortex and extensible by a C programmer; and builtin functions, also linked into Vortex. Functions are called by giving the name in a tag, possibly followed by arguments as attributes:

```html
<banner>
```
When the function ends, script execution resumes at the statement following the call.

The function `main`, which must be defined in all Vortex scripts, is the default start function of the script: execution begins with a call to it. (A different start function can alternatively be given in the URL; see p. 8.)

### Script Functions

Script functions are declared with HTML anchor (`<A>`) tags:

```html
<A NAME=banner>
  <HEAD><TITLE>Search the site</TITLE></HEAD>
  <BODY BGCOLOR=white>
</A>
```

The `<A NAME=banner>` tag opens the function, and a matching `</A>` close tag ends it; in between are the function’s statements. Optional parameters would be specified after the `NAME` attribute.¹ For full details on script functions and parameters, see function declarations, p. 12.

All text in a script, whether Vortex commands or HTML, must be contained within script functions. The exceptions are comments (delimited by `<!` and `>`, e.g. `<!-- comment -->` or `<!DOCTYPE>`²) and Vortex directives such as `DB` and `EXPORT`.

### User Functions

User functions are built into Vortex itself, and available to any script. New ones can be added by a C programmer. They are called like script functions, with the difference that arguments are not named and hence order is significant:

```vortex
<substr $str $offset $len>
```

See p. 564 for details on creating user functions in C.

### Builtin Functions

Builtin functions are generally called like user functions, and are also available to all scripts.

### 1.1.3 Statements

Functions are composed of HTML and Vortex statements. Any text or tag which is not recognized as a Vortex statement, directive, variable, comment or function call is simply printed out. For example, the `HEAD` and `TITLE` tags in the `banner` function above are not Vortex statements, so they are printed.

¹Script function parameters are available only in Vortex versions 2.6 and later.
²While not strictly an HTML comment, this has historically been interpreted as a comment by Vortex.
To aid in code formatting, any leading whitespace on a literal line of HTML is stripped in the output. This allows proper indentation of source code in the Vortex script, while reducing redundant space in the output. In fact it can lead to a significant reduction in the size of the output, increasing response time for users on slow connections. (This stripping does not happen within VERB elements; see p. 56.)

### 1.1.4 Variables

A Vortex variable is denoted by a dollar sign ($) immediately followed by a name. Variable names have the same syntax as function names (p. 12), i.e. alphanumerics, underscores, spaces or a period, and are case-sensitive (i.e. $value and $Value are distinct variables). If a variable name contains non-alphanumeric characters, or is adjacent to text not part of its name, its name must be quoted to delimit it, e.g.:

I $'prefix'opened the case.

A variable is printed by simply referring to it in the Vortex script, as above. Its value is automatically HTML-escaped when printed (unless the script content type is non-HTML). To print a literal $, use $$:

The price is $$39.95.

### Variable Values

The value of a variable is an array of zero or more values. A reference to a variable will normally print its 0th (first) value, if any. Inside a LOOP or SQL statement, however, only the current value is used. This simplifies iterating a list of values, such as when creating checkboxes for a form (see p. 25 for details).

Variables can be assigned values in many ways, such as with an assignment statement (p. 21); as the result rows of a SQL statement (p. 27); or as the return value of builtin/user functions (e.g. $ret).

### Variable Types

A variable’s type is what kind or style its values are. A variable may have any Texis SQL type, e.g. varchar for a simple text string, int for integers, or counter for a counter field. The type of a variable depends on how it was assigned. Returned variables from SQL statements (p. 27) have the type of the corresponding table column. An assignment statement (p. 21) determines the type of the assigned-to variable from that of its arguments. The type of the return value of user/builtin functions depends on the function.

Where needed, a variable’s values are cast, i.e. copied and changed to, a different type. For example, passing an integer variable to a SQL statement that expects a varchar field, will cause the integer value to be cast to a string. This casting is usually done implicitly; in some rare cases, however, it may not be possible and an error results (e.g. a varbyte type cannot be cast to a date type).

A variable’s type affects how it is used in expressions, most notably comparisons. Integer values are compared numerically, so the integer 123 is greater than the integer 45. Strings (varchar type) however,
are compared alphabetically, so the string value “123” is *less* than the string “45”, just as “abc” is less than “def”.

**Variable Scope: Global vs. Local**

A variable’s *scope* is the range of the script where it is visible. Variables have either *global* or *local* scope. A global variable exists only once in a script, and is visible in every function. Modifications to it in one function are permanent and visible to all functions. Unless declared otherwise, all variables in a script are global. Global variables are useful for values that are relatively constant, or that many functions in the script must access, such as a session id.

A local variable, however, has a limited scope: it exists only within the block that it is declared in. Once that block ends, the variable is destroyed and its values lost. A local variable of the same name declared elsewhere is a different variable. A local variable can even exist multiple times simultaneously, if its block is entered again before it’s exited – i.e. a recursive function call. Each call of the function will have a distinct local variable.

Local variables must be explicitly declared, either as parameters to a script function (p. 12), or with the `LOCAL` statement (p. 23). They are used to clearly pass parameters to functions, or as temporary “scratch space” for a function without the side effects of global variable modification.

A local variable with the same name as another in-scope variable will have precedence over the outer variable. For example, a local variable named `$x` has precedence over, and will “hide”, a global variable named `$x`:

```plaintext
<A NAME=main>
  <$x = "fragile"> <!-- global $x -->
  <LOCAL x>
    <$x = "test"> <!-- only local $x seen here -->
  ...  
  </LOCAL>
  The value of x is $x <!-- back to global $x -->
</A>
```

Within the `LOCAL` block, references to `$x` refer to the local variable: the global `$x` is hidden. Thus, the global `$x` value “fragile” is not lost when the `LOCAL` block ends.

**Variable Precedence and Initialization**

When a script starts, the initial values for global variables come from one of several sources, in order of decreasing precedence:

- URL state
- EXPORTed variables from the URL.

---

3Local variables are available only in Vortex versions 2.6 and later.
4Except for reference arguments; see p. 15.
• State table
  EXPORTed variables from the state table.

• Command line
  Variables initialized on the command line.

• Environment
  Environment variables set by the server, such as $REMOTE_ADDR.

• HTTP cookies
  The value of any HTTP cookies sent by the Web client.

• URL
  Ordinary URL variables given in the query string of the URL (i.e. via the GET method on a form, or EXPORT QUERY variables).

• Content
  Query-string style variables given in the content (i.e. the POST method on a form). These could also be multipart MIME variables from a form upload (p 6).

These sources are checked in decreasing order: the first source that has values for a given variable will be used. Only values from this first (highest precedence) source will be used to initialize the global variable. If none of these sources has a value for the variable, it has no initial values.

Because of this ordering, lower-precedence values cannot override higher ones when more than one source has values for the same variable. An EXPORTed state variable will always have its state value rather than the value from a form if both are given; the form value is only used if no state was saved. An environment variable like $REMOTE_ADDR will be used rather than a URL variable of the same name. This helps prevent (un)intentional manipulation of a script’s variables by the Web client: more “trusted” sources are used before less “trusted” ones.

This precedence ordering can be altered in EXPORTed variables by using the USEROK flag (p. 62). When an EXPORT variable is flagged USEROK, the URL and Content sources (i.e. form variables) have highest precedence, before the state is checked.

Variables from Multi-part File Uploads

In addition to URL-encoded variables from forms and the URL, Vortex also imports variables from a form-based multi-part file upload, if the user’s browser supports them. These are variables of INPUT type file on a FORM. Note: the form encoding type must be explicitly set to “multipart/form-data” in order for most browsers to properly send the data:

```html
<FORM METHOD=post ACTION=$url/func.html
  ENCTYPE="multipart/form-data">
  File to upload:  <INPUT TYPE=file NAME=image ACCEPT="image/gif">
  <P>
  Description:  <INPUT TYPE=text NAME=desc>
  <P>
```
1.1. OVERVIEW

Action:

    <INPUT TYPE=submit VALUE="Submit">
    <INPUT TYPE=reset VALUE="Reset Form">
</FORM>

This form would present the user with a file-select dialog box (for the image variable) and a plain text box (for the desc variable). On submission, the browser sends the file the user selected, and the Vortex variable $image would contain the contents of that file. Any other type variables on the form (e.g. desc) are imported as usual.

Debug Syntax

To quickly print information about a variable when debugging, it can be printed with a question mark after the dollar sign, e.g. $?myVar. This will print the variable name, type and all in-context values. In HTML mode, styles and colors are applied to distinguish the output and values from one another; for example, empty values have a light-green background. Even more debug information can be printed with two question marks, e.g. $??myVar, such as script and line number, variable scope, and referenced source variable. Both syntaxes are legal only when printing, i.e. not for parameter passing etc. The format and content of the output is intended for short-term debugging, not production scripts, and may change in a future release. These syntaxes were added in version 6. See also <varinfo dump>, p. 278.

In Vortex version 7.03 and later, one of two additional flags may be given after the single or double question mark:

- + (plus sign)
  Forces HTML-escapement of variable values, regardless of HTML mode.

- - (minus sign)
  Forces as-is printing (no HTML escapement) of variable values. By default, HTML escapement happens if the script is in HTML mode; using this flag may be useful to avoid escapement when an HTML-mode Web script is being debugged to a log file that is being read as plain text.

1.1.5 Execution

Vortex scripts are specified in the URL to the texis CGI program (see below), or as an argument if run from the command line (p. 567). In either case, the script is first compiled to an internal form, which is written out to a file with the same name and a .vtx extension.

The P-code (.vtx) file is run on each invocation, which is much faster than re-interpreting the script every time. Subsequent invocations will automatically recompile the script if its source is changed. Any errors encountered during compile or execution are printed out, with the script name and line number if applicable. The errors are also logged to the vortex.log file (see p. 581 for the format). These errors may be trapped and handled by the script during execution with a putmsg function.

Script execution starts with the function given in the URL. If no function is specified in the URL, then the value of the CGI variable cmd (section 1.4.1) is used. If neither is given or is an invalid or PRIVATE/EXPORT function, execution defaults to the main function.
1.1.6 URL Syntax

CGI Mapping by Directory

When the web server’s CGI environment is mapped from programs in a specific directory (e.g. /cgi-bin), Vortex scripts are specified in URLs as a path appended to the texis program URL /cgi-bin/texis. With most CGI programs, this path (the PATH_INFO CGI variable) is interpreted as relative to the server’s document root (i.e. where HTML documents are). However, Vortex uses its own ScriptRoot directory instead of document root, to avoid file permission issues. ScriptRoot defaults to the texis/scripts subdirectory of the install dir (see p. 576 to re-configure).

Thus, if Vortex is installed such that its URL is /cgi-bin/texis, the URL /cgi-bin/texis/myVortexScript would execute the Vortex script /usr/local/morph3/texis/scripts/myVortexScript (standard Unix install) or C:\morph3\texis\scripts\myVortexScript (standard Windows Texis install).

CGI Mapping by File Extension

When the web server’s CGI environment is instead mapped by the Vortex source (.vs) or object (.vtx) file extensions, the Vortex interpreter program texis may not be present in the URL, and the Vortex script must have the .vs or .vtx extension. The remainder of the URL syntax is the same. For example, the URL /myVortexScript.vs would execute the Vortex script /usr/local/morph3/texis/scripts/myVortexScript.vs for Unix or C:\morph3\texis\scripts\myVortexScript.vs for Windows.

Full Syntax

The full syntax of a Vortex URL is (optional parts in []):

/cgi-bin/texis/myVortexScript[/+state][/function.mime][/+/userpath]

/cgi-bin/texis is the CGI path of the Vortex executable (if CGI mapping by directory).
/myVortexScript is the (ScriptRoot-relative) path of the Vortex script. It must not contain any path elements that begin with a + sign (due to possible confusion with other parts of the URL syntax). In versions prior to version 6, it should not contain a file extension. In version 6 and later, the Vortex source (.vs) or object (.vtx) extensions may be present (e.g. for CGI mapping by file extension).

If no script path is given, the Vortex script index in the default database (see p. 576) is run, if it exists and Vortex was invoked in the CGI environment. This allows a default script to be run if Vortex is invoked without a path.

If the script is EXPORTing any variables (see p. 61), then encoded state information may appear as another “filename” after the script name (i.e. the /+state part). This information is generated automatically by

---

5Some licenses require that the URL have the word “texis” visible in the path for Vortex to run.
6In Vortex versions prior to 5.0, the server’s document root was always used, instead of ScriptRoot.
7A Vortex file extension is optional when the web server is configured to map CGI by directory instead of by file extension.
the $url variable (see p. 87), which will contain all of the URL up through and including any state
information. Only the function, MIME extension and user path, if desired, need be appended by the
programmer in scripts.

The optional function name and MIME extension come next. If these are given, the function name becomes
the start function of the script, instead of the default main. This makes it easier to re-enter a script on
subsequent invocations, without cluttering up main with logic to call the right function. A non URL-safe
function name (e.g. containing spaces or /) must be URL-encoded.

The .mime extension is an extension which indicates the MIME type to set in the Content-Type output
header, e.g. .gif for “image/gif”. It also provides a clue to Web browsers and end users what type of
information the URL will return: a .gif URL is expected to yield a GIF, etc. The default if no extension is
given is HTML.

Note that the function name and MIME extension, if given, must both be present. The MIME extension
should be one of the following recognized extensions:

.au, .bmp, .css, .doc, .dvi, .eps, .gif, .gz, .htm, .html, .ief, .jpe, .jpeg, .jpg,
.laTeX, .mov, .mpe, .mpeg, .mpg, .pbm, .pdf, .pnm, .ps, .qt, .ram, .ras, .rgb, .rtf, .snd, .tex, .textinfo, .tiff, .txt, .wbmp, .wml, .wmlc, .wmls,
.wmlsc, .xbm, .xls, .xml, .xpm, .xwd, .Z, .zip

If the MIME extension is given but not one listed above, the Content-Type will be
“application/octet-stream”. (Note: this list can be altered in the Texis Web Server; see the
TypesConfig setting (p. 590.)

The MIME extension bin is special: if it is given, then no headers are printed by Vortex. Normally, a
Content-Type and other HTTP headers are printed by Vortex at the start of execution. When the MIME
extension bin is used, however, the Vortex script is responsible for correctly printing all HTTP headers.
Note that in Vortex version 2.6.913000000 19981207 and later, the header function (p. 144) is preferred
for handling headers, as it is more flexible.

The /userpath comes last, though it is rarely used. At the end of the URL (i.e. after $url and
function/MIME info), “/+” followed by an arbitrary, user-defined absolute path may be given. This path
will be assigned to the special variable $userpath at script start. This syntax provides a way to attach an
arbitrary path to the URL, much like $PATH_INFO does for ordinary CGI programs (but $PATH_INFO
has been usurped for Vortex use). It is only used, in combination with server aliases, in circumstances where
the entire URL must truly “look” like a path. The user path, if given, must start with “/”.

Query-string variables may also be appended to the URL, after a “?” character. These can either be set
explicitly, or indirectly from EXPORT QUERY variables with $urlq (see p. 61).

The URL syntax is designed to keep the information it holds, i.e. script name, state information, start
function and MIME type, in a logical format. To Web users viewing the URLs, everything appears as a “file
tree” beneath the Vortex executable. Each function is a “file” in the “directory” of its script.

Examples

The simplest URL is just that generated by $url in the script:
See <A HREF=$url>here</A>.

This URL might look something like this to the user (if they view the resulting HTML source in their browser):

See <A HREF=/cgi-bin/texis/myVortexScript/+wwr4tyuq>here</A>.

When this URL is invoked by the user, the same script (/myVortexScript) is run. The script's EXPORTed variables, having been automatically saved (encoded in /+wwr4tyuq), are restored to their values. No function/MIME extension was given, so the script defaults to starting at main and content type text/html.

A more complex script may need to start at a particular function. For example, a search script may print the results of a search, and give a URL next to each hit for more information on that particular item. These URLs, when followed, would call a function like details for further display of the chosen item. To accomplish this, the details function name is simply appended to $url in the script, along with a MIME extension, when printing the initial search results:

For more info, see <A HREF=$url/details.html>here</A>.

When this URL is invoked, the script will start at the function details, and set the Content-Type to “text/html”.

The MIME extension may differ if a different content type is needed. A script with a link to a print function that generates a Postscript document might use this:

<A HREF=$url/print.ps>Print</A> this document.

Here the function print is the start, and the MIME type application/postscript (instead of text/html) will be set, which tells the user's browser to treat the document as Postscript.
1.2 Vortex Statements
1.2.1 **A – function definition**

**SYNOPSIS**

```html
<A NAME=name [PUBLIC | EXPORT | PRIVATE] [param [=value|$var] ...]>
... statements ...
</A>
```

**DESCRIPTION**

An anchor (`<A>`) tag delimits Vortex script functions. The function name is given by the `NAME` attribute. Following the anchor tag are the function’s statements, and a closing `</A>` tag ends the function.

Function names, like variable names, are composed solely of alphanumeric characters, underscores (`_`), periods or spaces (i.e. if the name is in quotes), optionally starting with a slash (`/`). There must be at least one letter or underscore, which must occur before any digits. A function name cannot be a reserved word, such as an HTML or Vortex tag, or the name of a previously-declared function (such as a built-in or user function). These are the reserved tag names in Vortex:

- `ADDTRAILING
- `DEFAULT
- `FMTINFO
- `PUTMSG
- `SQL
- `TRAP
- `ARRAY
- `ELSE
- `GETVAR
- `READLN
- `SQLCACHE
- `UNIQ
- `BREAK
- `ELSEIF
- `HASH
- `READVARS
- `STACK
- `USER
- `CAL
- `ENTRYFUNC
- `HEADER
- `RETURN
- `STAT
- `USES
- `CALDATE
- `EXEC
- `IF
- `REX
- `STRFMT
- `VARINFO
- `CALRULE
- `EXIT
- `LOCAL
- `SB
- `SUBMIT
- `VXCP
- `CAPTURE
- `EXITFUNC
- `LOOP
- `SCHEDULE
- `SWITCH
- `WATCHPATH
- `CASE
- `EXPORT
- `MM
- `SETVAR
- `TIMEOUT
- `WHILE
- `CONTINUE
- `FETCH
- `NSLOOKUP
- `SORT
- `TIMPORT
- `WRITE
- `COOKIES
- `FMT
- `PAGELINKS
- `SPLIT
- `TRACESQL
- `XTREE
- `DB
- `FMTCP
- `PASS
```

Script function names are case-sensitive. A function name may start with a slash (`/`), so that end-tag like functions can be declared. Function declarations cannot be nested a la Pascal; any anchor tags inside a function are printed out like any other non-Vortex tag. However, unlike some languages, Vortex functions can be called before the point they are declared (i.e. no C-like prototypes are needed).

**Function Scope**

The scope of a function – where it is “visible” and may be called from – can be altered with one of the following attributes after the `NAME` attribute in its declaration, in decreasing order of visibility:

- **PUBLIC**
1.2. VORTEX STATEMENTS

A PUBLIC function is visible everywhere – to the file it is declared in, to other linked-in modules or scripts (see p. 558 for a discussion of library modules), and to users, i.e. it may be the start function for a script.

- EXPORT
  An EXPORT function (not to be confused with the EXPORT directive, p. 61) is visible to the file it is declared in, and to other linked-in modules or scripts. However it is not visible to the user, and therefore cannot be the start function. The EXPORT attribute is used in library modules to make sensitive functions available to other scripts but not to the outside world. The EXPORT attribute is available in version 2.6.936300000 19990902 and later.

- PRIVATE
  A PRIVATE function is visible only to the file it is declared in. It cannot be a start function, nor can other linked modules or scripts see it. Indeed other modules could redeclare their own distinct function with the same name.

An attempt to call a function outside its scope will have the same result as if the function doesn’t exist. For example, trying enter a script at a PRIVATE or EXPORT function will start at main instead. PRIVATE functions provide a measure of security by preventing web users from entering a script at an unintended point. For example, a function such as this:

```html
<A NAME=deluser PRIVATE>
  <SQL NOVARS "delete from users where User = $User">
  </SQL>
  User $User was deleted.
</A>
```

could be dangerous if invoked by the user at a point not controlled by the script: the $User variable might not have been verified. For similar reasons, all user and builtin functions are inherently PRIVATE. However, the script function main must always be PUBLIC, as it is the default start point.

If a function does not have its scope declared, Vortex will try to default it to PRIVATE, as an additional security measure. However, this is not always possible, for back-compatibility reasons. Thus it is wise to declare explicitly the scope of all functions, and to use the lowest scope possible (e.g. PUBLIC only if specifically required). A function is PRIVATE if the following is true:

- It is explicitly declared PRIVATE, or
- It has parameters, or
- One or more other functions in the script have their scope declared explicitly

otherwise it is PUBLIC. These arcane rules maintain back-compatibility with Vortex versions prior to 2.1.895000000 19980513, where all script functions were PUBLIC (and had no parameters). Again, it’s easier to simply always declare function scopes explicitly.
Function Parameters

Script functions can be declared with parameters, which are passed to the function when it is called. Parameters are declared by naming them in the `<A NAME>` declaration tag, after the `NAME` attribute and any scope attribute (PUBLIC/PRIVATE etc.). Each parameter is a local variable (p. 5) inside the function:

```html
<A NAME=banner title="Untitled" bg>
  <HEAD>
    <TITLE>$title</TITLE>
  </HEAD>
  <BODY BGCOLOR=$bg>
    <CENTER>
      <H2>$title</H2>
    </CENTER>
  </BODY>
</A>
```

This function has two parameters, `$title` and `$bg`. When the function is called, its arguments are named in the tag (like HTML attributes):

```html
<banner title="Search Results" bg=white>
...<banner bg=white>
```

Since the arguments are named, they need not be given in order (like HTML). Nor are all parameters required when calling: the `$title` parameter was declared with a default value of "Untitled", so it will have that value if unspecified in a call:

```html
<banner bg=white>
```

Parameters without default values (e.g. `$bg`) are required in calls, however. This provides the ability to force their setting in situations where a default value might not be useful. For example, a search-and-replace type function could require its search and buffer parameters: it may not be sensible to search a default buffer. The exception to this is start functions: since they have no explicit "caller" to name arguments, no-default parameters in the start function are initialized like a global variable, i.e. from the appropriate environment, query, POST etc. variable of the same name.

The default value for a parameter can also be a variable:

```html
<A NAME=banner title=$DefaultTitle bg>
  ...
</A>
```

In this case, if `$title` is not specified in a call, it will have the value of the global (never local) variable `$DefaultTitle`.

---

8 Script function parameters are available only in Vortex version 2.6 and later.
9 Unlike user/builtin function calls, where arguments are unnamed and passed in declared order.
1.2. VORTEX STATEMENTS

Pass-by-Value Arguments

A function’s arguments are passed by value by default. This means that each parameter receives a copy of its argument’s value(s), so modifications to the parameter do not affect the caller’s argument – even if the variables are the same name – and changes are lost at function exit. For example, here the function `twiddle` modifies its parameter $x$:

```vortex
<A NAME=twiddle x>
 $x = (x + 1)$
</A>

<A NAME=main>
 $x = 3$
 x starts at: $x$
 <twiddle x=$x$>
 x is now: $x$
</A>
```

When `twiddle` is called in `main`, it is passed a copy of the global variable $x$. Upon return, the global $x$ in `main` will still be 3: `twiddle` modified a local $x$ that was discarded.

This behavior assures a caller that its local arguments will not be modified by the function without the caller’s knowledge. (It also allows a function to modify its parameters as “scratch space”.) Also, unlike some languages, there is little overhead in passing very large-valued arguments by value, because Vortex uses copy-on-write buffers for variables. Internally, variable data is copied only as needed, not just every time a function is called.

Pass-by-Reference Arguments

Sometimes it is desirable for a function to intentionally modify its arguments, perhaps to return multiple value lists. In such cases the argument can be passed by reference with the `&` variable syntax:

```vortex
<A NAME=square x>
 $x = (x \times x)$
</A>

<A NAME=main>
 $y = 5$
 <square x=$&y$>
 y squared is: $y$
</A>
```

Here the $y$ argument is passed to $x$ by reference. A reference argument is not copied to its parameter; instead the parameter becomes an “alias” for the caller’s argument. Any modifications to the parameter in the function will change the caller’s argument too: the two are the same variable. Thus, when $x$ is
modified in the square function above, $y$ in main is changed as well, since it was passed by reference when square was called.

Note that unlike other languages, a pass-by-reference argument is determined by the caller, not the function. I.e. the function declaration has nothing to do with whether an argument is a reference or not; it depends on how it is called. Thus, unlike C++ references, the caller can always control whether its arguments are modifiable or not, even if the declaration of the function itself is unknown or changes.

**Start Functions with Parameters**

If the start function to a script has parameters, they will be passed as arguments from the same-named global variables if set, otherwise their default values/variables are used.

**CAVEATS**

The **PUBLIC** and **PRIVATE** flags were added in version 2.1.895000000 19980513. **EXPORT** was added in version 2.6.936300000 19990902. Function parameters were added in version 2.6.911900000 19981123.

All `<A>` tags in a function must be balanced with closing `</A>` tags, so that the function-closing `</A>` tag can be determined. If a function has unbalanced anchor tags, they should be printed in a `<VERB NOESC>` element (p. 56) to escape their meaning in Vortex.

Script functions generally default to **PRIVATE** if undeclared; however this is not always true, so all functions should be explicitly declared.

Arguments are named in script function calls, whereas in user function calls arguments are merely listed in order.

Even with pass-by-reference, changes to a parameter inside a function might not affect the caller’s argument, if a later call uses pass-by-value instead (i.e. it’s up to the caller, not the function).

User functions cannot pass arguments by reference.

A parameter hides a global variable of the same name from within the function: the global cannot be accessed (unless it was passed as a reference).

**SEE ALSO**

Built-in functions, user functions
1.2. VORTEX STATEMENTS

1.2.2 IF, ELSE, ELSEIF – conditional execution

SYNOPSIS

<IF condition1>
  ... statements if condition1 true ...
[<ELSE[ ]IF condition2>
  ... statements if condition2 true ...
][<ELSE>
  ... statements if both are false ...
</IF>

DESCRIPTION

The IF statement evaluates the given conditional expression. If the result is true (i.e. nonzero), then the
statements following the <IF> tag are executed. If the result is false, those statements are skipped and
control falls to the next statement after the closing </IF> tag.

An optional ELSE clause may be present, in which case the statements following the <ELSE> tag are
executed instead if the result is false. If the ELSE clause is another IF statement, it can be made part of the
ELSE tag by using <ELSEIF>. Thus, the left and right statements below are equivalent:

<IF $x eq 5> <IF $x eq 5>
  X is five.   X is five.
<ELSE> <ELSEIF $y gt 10> <IF $y gt 10>
  Y is greater than 10.   Y is greater than 10.
<ELSE> <ELSE>
  X is not five and Y is <= 10.   X is not five and Y is <= 10.
</ELSE> </IF>
</IF>

If many ELSEIF clauses are used on the same value, a SWITCH statement (p. 19) may be a clearer
alternative.

The condition of an IF statement can be simple or complex. A simple condition is of the form arg1 op
arg2, where each arg is a variable or literal string, and op is one of the following operators:

- eq or =: Equals
- ne or neq: Does not equal
- gt: Greater than
- ge or gte: Greater than or equal to
• \texttt{lt}: Less than
• \texttt{le} or \texttt{lte}: Less than or equal to
• \texttt{nmod}: Not mod; true if \texttt{value1 mod value2} is 0

For example:

\begin{verbatim}
<IF $temp lt 32>
  Ice
<ELSE>
  Water
</IF>
\end{verbatim}

A SQL expression condition can be any valid Texis SQL \texttt{WHERE} clause. Thus, \texttt{and} and \texttt{or} may be used, or even like:

\begin{verbatim}
<IF $limit lt 5 and ($ans eq "correct" or $body like $query)>
  You're under the limit, and either
  have the correct answer or the text matches your query.
</IF>
\end{verbatim}

Caveats

Variables should not be embedded inside a literal string, e.g. \texttt{"this is a $test."}. Literal string values should always be quoted, or they may be misconstrued as non-existent SQL columns.

The results of some operators can depend on the original type of the values, e.g. comparison of integer vs. string values (see discussion under Variable Types, p. 4).

If a variable contains more than one value in its current context (i.e. outside a \texttt{LOOP}), only the 0th (first) value is used. An empty (no values) variable is considered an empty string.

See also

\texttt{SWITCH}
1.2. VORTEX STATEMENTS

1.2.3 SWITCH – multi-choice branch

SYNOPSIS

```xml
<SWITCH switchValue>
  <CASE value1[ /]>
  ... statements if switchValue = value1 ...
  [</CASE>]
  <CASE op value2[ /]>
  ... statements if switchValue op value2 is true ...
  [</CASE>]
  ...
  <DEFAULT[ /]>
  ... statements if no <CASE> matches ...
  [</DEFAULT>]
</SWITCH>
```

DESCRIPTION

The SWITCH statement provides a multi-option branch based on a given switchValue, which is a single variable or literal. The value of switchValue is compared against each CASE value, in the order they appear in the script, until one matches. The statements that follow the matching CASE are executed, up to the closing </CASE>, or next <CASE>, <DEFAULT>, or closing </SWITCH> tag. Only the first matching CASE (if any) is executed. If none of the CASEs match, then the statements after the <DEFAULT> tag, if present, are executed.

An optional operator can be given in a CASE to use in comparing its value to the SWITCH value, instead of the default eq. This can be useful in checking a range of values; note the order of CASE statements here:

EXAMPLE

```xml
<SWITCH $temp>
  <CASE lte 32> Ice
  <CASE lt 60> Cold water
  <CASE lt 100> Warm water
  <CASE lt 212> Hot water
  <DEFAULT> Steam
</SWITCH>
```

The same operators that are translated in <IF> (p. 17) are supported in <CASE>.

CAVEATS
Similar to IF. Note that </CASE> is not needed; execution doesn’t fall through to the next CASE as in C. The DEFAULT clause, if present, must be last, since CASEs are checked in order of appearance.

SEE ALSO

IF ELSE ELSEIF
1.2. VORTEX STATEMENTS

1.2.4 Variable assignment – assign values to variables

SYNOPSIS

<$var = [value1 ...] >  
<$var = (SQL expression)>

DESCRIPTION

Variables are explicitly set with an assignment statement. The variable on the left is set to the list of values on the right. The right-side values are either literals (plain strings) or other variables. Each value of a variable on the right is assigned to a single value of the left-side variable; values are not concatenated together. If no values are given on the right, the assigned-to (left) variable is unset (i.e. has no values).

Example:

<$code = "Klaatu" "barada" "nikto">  
<$said = "Gort" $code "right" "now">  

The value of $said would be the 6 values Gort, Klaatu, barada, nikto, right, now.

Literals on the right side of (non-SQL-expression) assignments are either varchar (string) or long. If the value has an optional leading minus sign, no leading zeroes, and is one to eight digits, it will be a long; otherwise it is varchar.

The left-side variable’s type will differ from the right if the types on the right are not identical. In version 7.02.1416623000 20141121 and later, if all values on the right are numeric (floating-point or integral types), the left-side type will be numeric, and just large enough to maintain precision and sign of all right-side values (if possible); otherwise the left side will be varbyte if the right-side types are all [var]char and at least one [var]byte is present; otherwise varchar is used. If any right-side type is var... or multi-value, the left-side will be var... Note that the type chosen might not be any of the right-side types; e.g. int and dword typically result in long, as the largest dword value cannot fit in an int without overflow. In previous versions, if the right-side types are not identical, the left side became varchar.

If the right side is enclosed in parentheses, then it is interpreted as a SQL SELECT expression and evaluated. It may then be any valid Texis select expression:

<$x = (19 * 6 + 5)>  
<$txt = ( "This is test " + $x )>

The value of $x would be the integer 119, and the value of $txt would be the string “This is test 119”; the + operator concatenates when given strings. Quotes and Vortex operators in SQL expressions are mapped as in IF (p. 17).
If the assigned-to variable is currently in a loop construct, e.g. inside a `LOOP`, `SQL` or `TIMPORT` that is looping over the variable, then only the current value of the variable is changed, to the first value of the right side\textsuperscript{10}. The type is cast to the overall type of the assigned-to variable. If no values are present on the right, then the assigned value is the empty string (“”).

**CAVEATS**

See type caveats above.

If a SQL expression (in parentheses) is being used, multi-value variables are converted according to the current `<sqlcp arrayconvert>` setting, and an empty (no values) variable is considered an empty string.

**SEE ALSO**

`SQL`, `EXPORT`

\textsuperscript{10}Modifying variables inside a loop is only valid in Vortex versions after Sep. 1 1997.
1.2. VORTEX STATEMENTS

1.2.5 LOCAL – declare local variables

SYNOPSIS

<LOCAL var[=value|$initValue] ...>
...
</LOCAL>

DESCRIPTION

The LOCAL statement declares variables to be local. Within its block, the named variables will exist as local variables (see Variable Scope discussion, p. 5), and will be destroyed when the block exits. They are distinct from, and hide, other variables of the same name (if any) currently in scope.

Local variables are useful as “scratch space” in functions, without the side effects possible with global variables (i.e. modifying another function’s information). Like script function parameters, they can be initialized with a value or default variable upon entry to the block. However, if the default is a variable, it is taken from the scope of the <LOCAL> tag: this means it could be another local variable, unlike function default variables which are necessarily global. LOCAL variables without default values are initialized to empty (no values).

The scope of LOCAL variables ends with a matching </LOCAL> tag. If a </LOCAL> tag is not given, the scope ends with the block enclosing the <LOCAL>.

EXAMPLE

<A NAME=sec2time sec>
<!-- Prints linear seconds value as hours, minutes, seconds -->
<LOCAL h m s>
  <$h = ($sec / (60 * 60))>
  <$s = ($sec - ($h * 60 * 60))>
  <$m = ($s / 60)>
  <$s = ($s - ($m * 60))>
  The time is $h hours, $m minutes and $s seconds.
</A>

This function uses local variables as scratch space in its computations, thus ensuring that it can’t modify the data of another function or global variables.

CAVEATS

The LOCAL statement was added in version 2.6.911900000 19981123.
Local variables are destroyed when their block exits and their values lost. The next time the block is entered, the variables will be re-initialized by the above rules.

A local variable of the same name as another in-scope variable will “hide” the outer variable from visibility.

SEE ALSO

Function definitions, Variable scope discussion
1.2.6 LOOP – iterate through variables

SYNOPSIS

\[
<\text{LOOP} \ [\text{SKIP}=s] \ [\text{MAX}=m] \ [\text{REV}] \ \$\text{var1} \ldots>
\]

\[
\ldots \ \text{statements} \ldots
\]

\[
</\text{LOOP}>
\]

DESCRIPTION

The LOOP statement iterates through the values of the given variables, executing the given statements each time. Inside the LOOP, any reference to a LOOP variable returns the current value of the variable, instead of the first (0th) value: in the nth loop iteration, the nth value of each variable is used. The loop continues until the last value of the variable with the most values is iterated, or MAX iterations have occurred. MAX is optional; its argument is an integer literal or variable.

If SKIP is given, then the first s values of the variables are skipped; i.e. iteration (and the value of $\text{loop}$) starts with the sth value instead of the 0th.

Inside the LOOP, the special variable $\text{loop}$ is set at the start of each iteration to the current loop count (i.e. the current array index). When the LOOP finishes, $\text{loop}$ is set to the number of iterations (plus however many values were SKIPped). This provides an easy way to enumerate or count values inside the LOOP.

The variable $\text{next}$ is set to the “next” loop value each iteration. Inside the loop, this is $\text{loop} + 1$: an easy way to count values starting from 1 instead of 0. When the LOOP finishes, $\text{next}$ is equal to $\text{loop}$: it is the value to use for SKIP in a new LOOP to continue iteration at the next value.

If the REV flag is given, then the variables are iterated in reverse order, from last to first values. However the values of $\text{loop}$ and $\text{next}$ are still incremented forwards.

If a BREAK statement is encountered inside the loop, the LOOP is exited at that point, as if none of the variables had any further values. The $\text{loop}$ and $\text{next}$ variables, however, will have whatever values they had inside the loop.

The single-value nature of iterated variables applies globally. If a function is called from within a LOOP, references to the current LOOP’s variables (if global) still return just the single current value. Variables not listed in the current LOOP tag, however, still return all their values inside a LOOP.

Modifying a LOOP variable inside the loop will only change the current value of the variable; see Variable Assignment (p. 21). This can happen not only with an explicit variable assignment, but indirectly if a function or statement is called that sets $\text{ret}$, for example. Thus LOOPing over $\text{ret}$ is generally not advised, as nearly any function called inside the loop will attempt to modify $\text{ret}$ – possibly losing type information and/or values. This is also true of other statements in Vortex that put variables in a loop, such as SQL. Use the texis.ini setting [Texis] Warn If Ret Loop or the --warn-if-ret-loop command-line option (p. 570) to warn when such code is compiled.

---

11 Modifying variables inside a loop is only valid in Vortex versions after Sep. 1 1997.
EXAMPLE
This example generates radio buttons for a list of colors (essentially the <radiobutton> function):

$$\text{colors} = \text{Red Orange Yellow Green Blue Violet}$$
$$\text{rgb} = \text{FF0000 FFA500 FFFF00 00FF00 0000FF EE82EE}$$
<LOOP $colors $rgb>
  <input type="radio" name="rgb" value="$rgb"/> $colors
</LOOP>

CAVEATS
The REV flag was added in version 3.0.947100000 20000105.

LOOPs can be nested; however the same LOOP variable should not be used in an inner LOOP since it is already a single value.

The values of “short” LOOP variables, i.e. ones with fewer values than others in the same LOOP, appear as empty when iterated past their end.

SEE ALSO
SQL, BREAK
1.2. VORTEX STATEMENTS

1.2.7 SQL – execute SQL statement

SYNOPSIS

<SQL [options] "SQL command" ["SQL fragment" ...][ /]>  
... Vortex statements ...  
[<SQL>]

DESCRIPTION

The SQL statement executes the given Texis SQL command (with a “;” appended if needed). The returned rows’ fields are assigned to the Vortex variables of the same name, one field per variable value. For each result row returned by the command, the corresponding variables are updated, and the statements inside the SQL block are executed.

The field variables returned by SQL behave just like LOOP variables (see p. 25), in that only the current value returned is visible inside the SQL loop. This allows easy iteration of result rows, yet retains all rows for future use.

With the default <sqlcp arrayconvert> settings, certain multi-value result types may be converted into more than one Vortex variable value per SQL row. For example, a single strlst result value that contains 3 strings will add 3 varchar values to its Vortex variable, instead of 1 strlst value. This conversion increases flexibility by avoiding the need for manual conversion of cumbersome types like strlst, but can cause result columns to become out of sync with each other, if some have more values added per row than others.

Flags/attributes that may be given before the SQL statement include:

- SKIP=n
  Skip the first n result rows. The skipped rows are not assigned to variables; the next (n + 1) row retrieved will be the first (0th) value of variables. This is typically used to skip to a specific “page” of results when paginating output.

- MAX=n
  Return at most n result rows. The default is all rows. Note that result rows are only counted after the initial SKIP, if any.

- NOVARS[=varname[,varname,...]]
  Do not return any field variables from the statement; just execute and loop over it. This is useful for insert and delete statements, which normally return the variable(s) from inserted/deleted rows, which may sometimes conflict with parameters or other variables. It can also save memory for SQL statements where many unneeded rows are returned. Note that $loop and $next are still set as usual, so rows can be counted. In version 6 and later, a CSV list of zero or more variable names may be given: if a returned column name is in the list, it is not returned/assigned, otherwise it is returned. NOVARS with no list (or an empty list) blocks all variables.

- OKVARS=varname[,varname,...]
Only return variables in the given list; columns not in the list are not returned/assigned to variables. Added in version 6.

- **ROW**
  Set return variables to a single row; do not append in a list. At each iteration, the previous row’s values are freed, and the return variables have a single value at the end of the loop. This is useful for SQL statements that return a large number of rows that are only needed one at a time: the script might otherwise run out of memory trying to retain all rows in the variables.

- **DB=db**
  Set the database to use for this statement. Normally the database used is the one last set with the `<DB>` statement (p. 37). However the DB option to SQL overrides this value for this SQL statement. This is useful for avoiding side-effects in SQL statements that work on an alternate database and don’t want to disturb the current `<DB>` value (which can be obtained with the vxinfo function, p. 292). Added in version 3.01.966300000 20000814.

- **USER=user**
  Set the username to access the database as. Normally taken from the `<USER>` statement value (p. 38); this option overrides it without disturbing it. Added in version 3.01.966300000 20000814.

- **PASS=password**
  Set the password to access the database with. Normally taken from the `<PASS>` statement value (p. 38); this option overrides it without disturbing it. Added in version 3.01.966300000 20000814.

- **NULL=value**
  Set the $null value for this statement. Normally taken from the $null variable; this option overrides it without disturbing it. Added in version 3.01.966300000 20000814. (See Parameter Substitution, p. 30). To set the null value to no-values, use NONULL.

- **NONULL**
  Set the $null value to empty (no values). Normally taken from the $null variable; this option overrides it without disturbing it. Added in version 3.01.966300000 20000814. (See Parameter Substitution, p. 30).

- **OUTPUT=format**
  Also output the results of the SQL statement in the specified format. Normally the NOVARS flag would be specified with this option, since the variables are now being printed. The currently known formats are:
  - **xml**
    Outputs the data in XML. Each field will be an element, named the same as the Vortex variable would be, nested in a `<result>` element per row, within a single `<results>` top-level element. Added in version 4.00.1001000000 20010920. In version 5.01.1226002400 20081106 and later, strlst columns have their individual string values printed in child `<value>` elements.
  - **xml:ado**
    Outputs the data in an XML format that is compatible with ADO’s XML persistence format. An ADO RecordSet can be opened with:
    ```
    rs.open URL,, adCmdFile
    ```
In general it is preferable to fetch the already formatted data and display that if you are using ASP. Added in version 3.0.974700000 20001120.

In addition, the following flags may be appended to the OUTPUT string:

- :utf8
  Assume data is ISO-8859-1 and translate it to UTF-8.

- :noutf8
  Do not translate data from ISO-8859-1 to UTF-8; assume data is already UTF-8. Illegal UTF-8 sequences, however, will still be assumed to be ISO-8859-1 and will be translated to UTF-8. This is the default in version 7 and later; previous versions defaulted to :utf8.

- :base64
  Data which contains bytes less than 32 or greater than 126 is encoded in base64. Enables output of binary data (for some XML processors which understand base64). Not supported in ADO format. Added in version 5.01.1100210584 20041111.

- PROVIDER=$provider
  Specifies what provider will execute the SQL. The value may be texis for normal internal Texis execution (which is also the default if unspecified or empty), or odbc to connect to an ODBC server to run the SQL. The odbc provider requires the CONNECTSTR option to be set, and is currently only supported in Windows versions of Vortex. Added in version 5.

  In version 5.01.1222808000 20080930 and later, wide character result data (e.g. nvarchar/SQL_WCHAR) returned from the ODBC driver is converted to varchar UTF-8, and hi-bit varchar parameter data (e.g. for INSERT) is converted from UTF-8 to SQL_WCHAR.

- CONNECTSTR=$connectstr
  Provides the ODBC connection string if PROVIDER is set to odbc. Added in version 5.

- PARAMPREFIX=$prefix
  Sets the Vortex variable parameter name prefix: an alternate method of passing SQL parameters, used only in exceptional circumstances. See Dynamic Parameters (p. 31) for details. Added in version 5.01.1226541000 20081112.

If a BREAK statement is encountered inside the loop, the loop is exited at that point, as if the SQL command generated no more rows.

CAVEATS

If a SQL statement that modifies a row is nested inside a SQL selecting from the same table then care may need to be used to ensure that the same row is not updated multiple times. This is most likely to occur when the outer SQL is selecting without an index, and the update may increase the size of the record. The size increase may move the record, and it may then reappear in the select.
Parameter Substitution

Variables can be placed in the SQL command as parameters. These variables are not merely substituted as strings, but become distinct arguments to the Texis SQL command, preserving their type. Thus, no escapement of special SQL characters like "'" or ";' needs to be done for parameters, and binary data such as images may be passed safely for byte fields. There is also no danger of the SQL command being modified by a rogue argument (aka “SQL injection”), e.g. a variable argument value like "; DROP TABLE customer" won’t end the SQL command and get executed (but see notes under SQL Command Construction, p. 31). Parameters that are multi-value variables may be converted into another type (e.g. an array of varchar values converted into a strlist), depending on the current $sqlcp arrayconvert> settings.

To simplify construction of complex WHERE clauses, parameter variables can be automatically dropped from a SQL SELECT query. Normally, all variables embedded in the SQL command become parameters; if a variable is unset (has no values) it’s treated as a single empty string (“”) parameter. If the special variable $null is set, however, any single-value parameter that matches any value of $null is dropped from the query, and its part of the WHERE clause collapses.

For example, in the following query $null is set to “any”:

```xml
<$xval = "any">
<$yval =
   "This is a test."
   "So is this."
>
<$null = "any">
<SQL "SELECT result
   FROM Text
   WHERE X = $xval AND Y = $yval">
   $result
</SQL>
```

Since the SQL parameter $xval has one value that matches $null, it is dropped from the SQL query and the WHERE clause becomes equivalent to “WHERE Y = $yval”. Unset parameter variables are treated as empty strings (“”) when comparing against $null.

This feature allows complicated queries to degenerate into simple ones when the extra variables are not needed, without cumbersome checking of all the parameters. A common use is in HTML forms, where there may be several search fields that are optional (e.g. an option checkbox, or subject and author text fields). By setting $null to the empty string (<$null = ">), any unset checkboxes or empty (unfilled) text fields from the form can be implicitly dropped from the SQL query.

Note that this feature only applies to SELECT clauses. This is to help prevent inadvertent deletion of too many rows. For example, a DELETE statement whose WHERE clause parameters are accidentally left empty by the user might otherwise delete the entire table, if $null is "". Thus, any unwanted parameters must explicitly be left out in non-SELECT statements. For security, the $null variable can only be set explicitly by a script; it is always initialized to no-values at script start regardless of form/environment/etc. variables.

12 In Vortex version 3.00.958700000 20000518 and earlier this feature may apply to non-SELECT clauses as well.
1.2. VORTEX STATEMENTS

Dynamic Parameters

In rare circumstances, the SQL statement – and thus the number of possible parameters – is not known at script writing time, or may change dynamically. For example, a script may be attempting to execute a series of SQL statements (perhaps read from a log file) where the number of parameters changes with each statement, and the parameters are encoded separately. Parsing and modifying each SQL statement to insert an arbitrary number of $-variables would be cumbersome.

Instead, the PARAMPREFIX option can be set. When PARAMPREFIX is non-empty, parameters are denoted in the SQL statement by embedded question marks ("\?") instead of $-variables, and their values come from the Vortex variables with the name prefix denoted by the value of PARAMPREFIX, with numbers appended starting from 1 for the first parameter.

For example, given the statement:

```<SQL PARAMPREFIX="myParam" "SELECT * FROM myTable WHERE x=? AND y=?">```

the Vortex variables $myParam1 and $myParam2 would be used for the corresponding x and y column ?-parameters. Thus, arbitrary parameters from encoded data could be set by declaring (with <local>) a block of myParamN variables, and sequentially assigning them from the encoded data with <setvar>.

Note that a ? embedded inside a SQL literal string (e.g. “select x + 'A question ?' from myTable”) does not denote a parameter.

Note: PARAMPREFIX is an awkward method of passing parameters in most circumstances; embedding $-variables is preferred, especially when the SQL statement is known. PARAMPREFIX is only used in rare situations, with truly unknown SQL statements.

SQL Command Construction

The SQL command is normally a single Vortex string literal whose embedded variables become parameters. However, in some cases it is necessary to dynamically alter the command in ways not possible with $null. For example, the table may need to be set at run time, yet SQL syntax does not permit it to be a parameter.

The command can be set at run time by passing one or more variables or literals to SQL, instead of a single string literal. In this case, the variable(s) and literal(s) are concatenated together and become the command. Any variables embedded in this concatenated string then become the command parameters. Thus, an arbitrary SQL command can be constructed at run time by the Vortex script, yet still be protected from rogue arguments sent by users (SQL injection):

```<A NAME=main>
  <FORM METHOD=post ACTION=$url/search.html>
    Search: <INPUT NAME=title> <BR>
    What: <SELECT NAME=what>
      <OPTION>Magazines
      <OPTION>Books
    </SELECT>
</A>```
Note: For security, the resultant SQL command should not contain any value(s) that are not explicitly generated by the Vortex script itself. Any user-supplied values should be parameters ($title in this case) or otherwise checked first ($what), so that the command is still protected from SQL injection via a variable.

In the above example, the command is the concatenation of the arguments to <SQL>. The first variable in the command, $tbl, will be part of the SQL command, rather than a SQL parameter, because it is not embedded in a string literal. Thus, we must ensure it does not contain rogue SQL code: the <IF> statement explicitly sets it to a known string. The second variable, $title, will be a SQL parameter, because it is embedded in a literal string. Thus, no matter what its value, it cannot affect the action taken by the <SQL> statement: it is safe to import directly from the user.

Multi-value Variables

With the default <sqlcp arrayconvert> and metaphornstrlistmode settings (on and equivlist, respectively), a string variable with multiple values becomes a parenthetical comma-separated list when used as a SQL parameter, so that any of the values match with the Metamorph like operator. For example, the following query:

The capitals are:
$$states = "AL" "MS" "GA" "FL"
<SQL "select capital from State where state like $states"> $capital </SQL>

would match any of the given states, because the $states parameter becomes “(AL, MS, GA, FL)”.
Complex Field Names

Some field names returned by SQL may be invalid as Vortex variable names. For example, an embedded Texit function call such as:

```
<SQL "select Title, text2mm(Text) from books where id = $id">
</SQL>
```

would attempt to return the fields `Title` and `text2mm(Text)` as Vortex variables. However, “`text2mm(Text)`” is not a valid Vortex name. Such invalid fields are silently discarded by Vortex; in order to return them they must be renamed in the SQL statement:

```
<SQL "select Title, text2mm(Text) result from books where id = $id">
</SQL>
```

The `text2mm(Text)` field is here renamed to `result`, a valid Vortex variable name.

Another common example is the `$rank` value set for each value in a `likep` query. The “$” is part of the actual SQL field name. In Vortex however, it must be escaped, and the field renamed:

```
<SQL "select $$rank Rank, Title from books where Title likep 'Norway fjords'">
...
</SQL>
```

The rank value is here returned in the Vortex variable `$Rank`.

Special SQL Variables

The `$loop` variable is set in a SQL statement as it is in a LOOP, though any `SKIPped` rows are not counted in it (i.e. it starts at 0 regardless of ‘SKIP’). At the end of the loop it is thus the number of rows returned by the SQL statement.

The `$next` variable is also set at every iteration: it is the number of `SKIPped` rows plus `$loop` plus 1. This is a convenient way to number hits counting from 1, and consistently across result pages with varying `SKIP` values. When the SQL loop ends, `$next` is set to the number of `SKIPped` rows plus the number of iterated rows: the value to use for `SKIP` in the next SQL statement, for the next result page.

When the SQL statement is first executed, at the start of the first iteration, the `$indexcount` variable is set to an estimate of how many total rows (i.e. as if no `SKIP` or `MAX`) the query will return. Its value is the number of matching rows found in the index(es) used by the query (if determinable). It is only an estimate, since further processing may eliminate some rows from the final result set. The last value of `$next`, in contrast, is always an exact row count (provided the loop was not exited early due to `MAX` or `BREAK`). However, since `$indexcount` is set at the start of the statement, it can be used to quickly indicate the
(probable) number of result rows without looping through them all to get $\text{loop}$. It is also not affected by the MAX value, as $\text{next}$ and $\text{loop}$ are:

```sql
<SQL SKIP=$\text{skip} \text{ MAX}=10
   "select Title from books where Title like $\text{query}" >
<IF $\text{loop eq } 0> <!-- print this before the first row -->
    First 10 hits out of a possible $\text{indexcount} \text{ hits}:
</IF>
$\text{Title}
</SQL>
```

Note that if no indexes exist, none are used by the query, or an index is used but in a way that precludes pre-counting (e.g. in “bubble-up” mode), then $\text{indexcount}$ will be 0.

In versions after 3.0.942800000 19991116, the $\text{rows.min}$ and $\text{rows.max}$ variables provide more information than $\text{indexcount}$ alone. They are set to the minimum and maximum number of total rows the query will return, respectively, or -1 and -2 (i.e. less than 0) if unknown. $\text{rows.max}$ is similar to $\text{indexcount}$: an upper limit to the result row count (ignoring SKIP and MAX). However, $\text{rows.max}$ is updated once it is possible to reflect the final (post-processed) count, or set to less than 0 if no index/row count information is available. $\text{rows.min}$ is a lower limit, but also contains a one-row look-ahead: if $\text{rows.min}$ is greater than $\text{next}$ at any time (inside the <sql> loop or after the end), then at least one more row is definitely available. This makes it easy to check when to print a “Next page” link.

The $\text{rows.min}$ and $\text{rows.max}$ variables are updated every iteration and at the end of the SQL loop. Once the total count is known exactly, both variables are equal. Until then, they provide a “window” that bounds the result count, which closes as rows are processed until the exact count is known. If the exact count is known initially – as in the case of a fully indexed query – then $\text{rows.min}$ will be equal to $\text{rows.max}$ from the first row onwards. This helps eliminate the guessing game of when $\text{indexcount}$ is accurate or not.

In version 6 and later, the $\text{sqlresult.returnedmin}$ and $\text{sqlresult.returnedmax}$ variables are set, similarly to $\text{rows.min}$ and $\text{rows.max}$. However, they are often more accurate and consistent, as their information is obtained directly from the Texis SQL engine, and thus are recommended over $\text{rows.min}$/\text{rows.max}.

Also in version 6 and later, $\text{sqlresult.matchedmin}$ and $\text{sqlresult.matchedmax}$ are set. These are similar to $\text{sqlresult.returned...}$, but are the min/max total rows matched – i.e. by the WHERE clause, before likeprows, GROUP BY etc. reduce the returned result count. For example, in a LIKEP query, a maximum of 100 top-ranked results are generally returned (the default for likeprows), yet many more results might have actually matched the query. Thus, $\text{sqlresult.returnedmin}$ may not exceed 100, whereas $\text{sqlresult.matchedmin}$ may be much greater. The $\text{sqlresult.returned...}$ variables can thus be used to compute pagination links, and the $\text{sqlresult.matched...}$ variables used to compute the total hits for the ... of N hits message.

Also in version 6 and later, $\text{sqlresult.indexcount}$ is set, similarly to $\text{indexcount}$, but is -1 (not 0) when unknown, to distinguish from 0 (known but no results).

The $\text{null}$ variable has significance for parameters, as noted above (see Parameter Substitution, p. 30).
EXAMPLE

```sql
<sql>
    $query = "John";
    $table = "select Name, Address, Phone from customer where Name like $query";
    foreach ($rows in $table) {
        $name = $rows[0];
        $address = $rows[1];
        $phone = $rows[2];
        echo $name, $address, $phone;
    }
</sql>
```

There are $loop matching customers.

CAVENDIS

Multiple-argument SQL commands were added in version 2.1.899200000 19980630. Nestable SQL statements were added in version 2.1.873500000 19980905.

Column variables returned by the SQL command are cleared first before the loop starts, i.e. previous values are lost. However, if no rows are returned by the command, then the variables are not cleared, since it is unknown what variables would be returned.

Since variables in the SQL command become Texit parameters, they are only permitted where parameters are allowed, e.g. field values, arguments to WHERE clauses, etc. In particular, the SQL command name (e.g. SELECT, INSERT) cannot be a variable. This is for security (see Parameter Substitution, p. 30).

Care must be taken when passing the SQL command as a variable instead of a string literal (see SQL Command Construction, p. 31) that the command cannot be abused by the user, e.g. all user variables should only be SQL parameters.

The $null variable is not inherited from the CGI environment; it must be explicitly set by the script. It is important to note that unset variables (no values) are treated as variables with a single empty-string value (""") for the purposes of SQL parameters and checking against $null.

The $indexcount variable is only an early estimate. If no index can be used to pre-count results, it will be 0. If post-processing of rows is required, it may be an overestimate.

Only variables that are parameters to a WHERE clause can legally be dropped when empty (or matching $null). All other variables in the SQL command (e.g. field values for INSERT) must be set.

While SQL statements may be nested, this is generally not necessary and can degrade performance considerably if misused. A SQL join statement, or SQL statements in series, are usually better options.
Assigning to a returned variable while inside the loop will only modify the current value of the variable.

An exception to the one-row look-ahead nature of $rows.min / $sqlresult...minis non-select statements: looking ahead can cause more than the desired rows to be deleted/inserted/updated, so the look-ahead is not done.

SEE ALSO

DB, USER, PASS, SQLCACHE, vxinfo, pagelinks, LOOP, BREAK, TIMPORT
1.2. VORTEX STATEMENTS

1.2.8 DB – set database path

SYNOPSIS

<DB = "path">

DESCRIPTION

DB sets the database path used by SQL statements in the Vortex script. It can be either a directive or a command. As a directive, it appears before the first function, and sets the starting database for the script. All SQL operations, including state-table information, are done with this database by default.

If the script needs to change databases at run time, DB can be given as a command, inside a function. This changes the database for all future <SQL> calls (but not for state-table information, which needs to be read before the script runs). However, because of the global nature of a directive, DB is generally only set at the top of a script as a directive.

Also, the database used by a single <SQL> statement may be changed with the DB option to <SQL> (p. 27), which overrides the database path for that statement only, without changing it for future statements.

If no DB directive or command has been executed by the time a database-accessing function is encountered (e.g. SQL), the default database is opened: /usr/local/morph3/texis/testdb (under Windows, c:\morph3\texis\testdb). In version 2.6.922900000 19990401 and later, this default can be set in the texis.ini config file (see p. 576).

EXAMPLE

<DB = "/usr/local/mydb">
<SQL "select Title from book">
  $Title
</SQL>

CAVEATS

The database given by DB is used by all SQL operations, including saving of EXPORT variables. If it is changed at run time, however, state info is still saved to the original (directive or default) database.

SEE ALSO

SQL, vxinfo
1.2.9 USER, PASS – set SQL user and password

SYNOPSIS

<USER = user>
<PASS = pass>

DESCRIPTION

The USER and PASS commands set the SQL user and password for accessing databases. These values will be used by future <SQL> statements, unless overridden by the USER and PASS options to a particular <SQL> (p. 27).

EXAMPLE

<USER = bob>
<PASS = robert>
<SQL "select Salary from employee">
 $Salary
</SQL>

CAVEATS

The USER and PASS commands should occur before any SQL statements.

For security, the arguments to USER and PASS should not be plain-text strings in a user-visible script.

SEE ALSO

SQL
1.2.10 WHILE – conditional loop

SYNOPSIS

<WHILE condition>
  ... statements ...
</WHILE>

DESCRIPTION

The WHILE statement loops over its statements as long as the given condition is true. $\texttt{loop}$ is set to 0 before the loop is started, and is incremented at the bottom of every iteration. The condition has the same syntax as for IF.

A BREAK statement, if encountered, will exit the loop.

EXAMPLE

This example prints 10 image tags, for a bar graph:

<WHILE $\texttt{loop lt 10}>
  <IMG SRC=/icons/bar.gif BORDER=0>
</WHILE>

CAVEATS

The WHILE statement was added Feb. 6 1997.

An infinite loop can be created, if the condition is never false. Similarly, if the condition is never true, the loop is never executed.

Unlike most other looping statements, $\texttt{loop}$ is set before the WHILE condition is first executed, so it always has the loop’s value and not its previous value when the condition is tested.

SEE ALSO

IF, BREAK
1.2.11 READLN – read file a line at a time

SYNOPSIS

<READLN [options] $file[ /]>  
... statements ...
[/READLN>]

DESCRIPTION

The READLN statement reads the given $file one line at a time, returning the line(s) in successive values of $ret. If $file is “-” (a single dash), the standard input is read instead. For each line read, the statements inside the READLN block are executed, and $ret is looped over, much like a SQL statement.

As in SQL, the special variables $loop and $next are set inside the READLN loop. At the end of the READLN statement, $loop is the number of iterations completed (e.g. the number of lines returned); $next is that number plus the initial SKIP if any.

A BREAK statement, if encountered, will exit the loop.

Options that may be given are:

- REV
  Read the file in reverse order, and start from the end. This is extremely useful in analyzing the latest information appended to a constantly-growing file, such as a web server log, without attempting to read the entire file (often impossible). Note that the START and END expressions are matched in line-read order, which means that with REV the END line will occur at or before the START line in the file itself.

- START=$expr
  Start returning lines with the first one that matches the REX expression $expr. By default lines are returned starting with the first in the file (or last if REV).

- END=$expr
  Stop returning lines when one matches the REX expression $expr; it will be the last line. By default the rest of the file (until MAXLINES reached) will be returned.

- MAX=$n
  Return at most $n lines, e.g. loop at most $n times. Note that this is not necessarily the number of lines actually read from the file: returned lines (and hence MAX) are counted after the application of the START expression and MAXLINES. If unspecified, empty or negative, no limit is imposed.

- MAXLINES=$lines
  Read at most $lines from the file. Note that this is not necessarily the number of lines returned (looped over); it is a limit to read before the START and END expressions are matched. E.g. if the START expression matches line 50, the END expression matches line 100, and MAXLINES is 75, only 26 lines will be returned: lines 50 through 75. MAXLINES is a “safety limit” in case the START or
END expressions are not found: otherwise a huge file might be read in its entirety looking for a
START expression that does not exist, before any line(s) are returned at all. If unspecified, empty or
negative, no limit is imposed.

- **ROW**
  As in SQL, don’t append lines into a list in $ret, but replace previous value each time. Saves
memory if a copious file is being read and the lines need be examined only once. Also enables
functions called inside the block to return multiple values for $ret, since $ret is then not a loop
variable.

- **SKIP=$n**
  As in SQL, skip the first $n returned lines. Note that this is counted after START.

**DIAGNOSTICS**

READLN returns a list of the lines read from the file (without newlines), or the last line read if ROW is set.

**EXAMPLE**

Last 100 lines of the web server log:
<READLN REV ROW MAXLINES=100 /usr/local/httpd/logs/transfer.log>
  $ret <BR>
</READLN>

**CAVEATS**

The READLN statement was added in version 2.1.901200000 19980723. MAX and $next were added in
version 2.6.938650000 19990929.

Note the distinction between MAX and MAXLINES: MAX applies to returned lines (i.e. loop iterations),
whereas MAXLINES applies to read lines, some of which may not be returned (e.g. before the START
expression).

Support for standard input was added in version 2.6.939250000 19991006. If the REV attribute is applied
when reading from a non-file standard input, READLN can consume a lot of memory, as the entire file must
be read in first. It is not recommended to read standard input in a CGI environment, as it may have been
already read for POST form variables.

**SEE ALSO**

read, spew, WRITE
1.2.12 WRITE – write to file

SYNOPSIS

<WRITE [APPEND] [OUTPUT] [SKIPONFAIL] [FLAGS=$flags] $file>
   ... output to file ...
</WRITE>

DESCRIPTION

The WRITE statement redirects the output generated inside its block to the given $file, truncating if it exists.

The $file argument may also be

- an empty string - output will be discarded (i.e. /dev/null or NUL redirect)
- “-” (single dash) - stdout (i.e. no-op)
- “--” (two dashes) - top-level stdout (i.e. "break through" all outer <CAPTURE>, WRITE etc. and print to output).
- “---” (three dashes) - prints to stderr; this was added in version 7.01.1394744000 20140313.

If the APPEND flag is given, the file is appended to instead of truncated. The OUTPUT flag acts as it does with <CAPTURE>, i.e. the output of the block is also output normally (ala the Unix tee command), in addition to being redirected to $file.

The SKIPONFAIL flag, when given, causes the code inside the <WRITE> block to be skipped on failure of the $file to be opened, instead of still being executed. This can be used in situations where the block code is entirely optional or redundant if the file-open fails, e.g. it is all logging code: it avoids the overhead of running code whose output will be discarded anyway. On the other hand, the default when SKIPONFAIL is not given – always continue into the block even on file-open failure – ensures potentially critical block code will always be run even if the file cannot be opened, e.g. if a mix of critical and logging code is inside the block.

The FLAGS argument value is a CSV list of zero or more of the other flags. It allows flags to be specified dynamically at run-time if needed.

DIAGNOSTICS

WRITE has no effect on $ret.

EXAMPLE

This function appends a log message ($logmsg) to a log file:
1.2. VORTEX STATEMENTS

The WRITE statement was added in version 2.1.901600000 19980728. The empty-string, “-” and “--” functionality of $file was added in version 4.00.1004670000 20011101.

The OUTPUT, SKIPONFAIL and FLAGS options were added in version 7. In versions prior to 7, SKIPONFAIL was effectively always on.

While the output of WRITE is buffered, simultaneous writes to the same file by different processes is discouraged, as there is no guarantee any process’s data will be completely written before the others’.

SEE ALSO

read, READLN
1.2.13 EXEC – execute program

SYNOPSIS

<EXEC [options] program [arg ...][ /]>  
... input to program ...
</EXEC>

DESCRIPTION

The EXEC command executes the given program, with arguments. Under Unix, each value of each given
argument becomes a separate command-line argument to the program. Under NT, a command line is built
from the arguments, space-separated, and it is up to the program to parse it into individual arguments.

The output of the statements inside the EXEC element will be piped to the standard input of the program. Its
standard output will become the value of $ret, each line being a separate value. In version 5 and later,
$ret.stderr will contain the standard error from the program (unless redirected by options),
$ret.err will contain a list of putmsgs created by the execution, and $ret.exitcode will contain
the exit code of the program (or the terminating signal as a negative number, if Unix).

The following options may be given before the program argument:

- **NOBR**
  Do not split the returned output into lines; leave intact as one value.

- **BIN**
  The returned $ret value is considered binary (e.g. varbyte). This is useful for image processing,
  where the output of the program might be a binary image. BIN implies the NOBR flag.

- **BKGND**
  Background the program: do not wait for it to finish executing once </EXEC> is reached. The
  program’s input is still taken from the <EXEC> block, but its output will be ignored and $ret will be
  the process id of the background program instead. This is useful for starting programs that can’t be
  waited for, or do not need to be, such as sending a mail message.

- **TOFILE=$file**
  Send the output of the program to the file named by $file, instead of collecting in $ret
  (foreground) or discarding (BKGND). If $file is “-” (single dash), the program’s output will go to
  the Vortex script’s standard output (aka stdout). If $file is empty, the program’s output will be
  discarded (i.e. /dev/null or NUL redirect). Added in version 4.0.1004580000 20011031. See
  caveat below.

- **DOMAIN=$domain**
  Sets the domain for USER. Added in version 5. Only supported under Windows. The default if
  unspecified or empty is taken from USER if present, otherwise “.” (local machine) is used. Ignored if
  USER is unset or empty.
1.2. VORTEX STATEMENTS

- **USER=$user**
  Specifies the user to login and run the program as. The default if unspecified or empty is the current user (no login). Added in version 5. Only supported under Windows. If **DOMAIN** is unspecified or empty, a domain may also be given in the value, e.g. Domain\User or User@Domain. PASS should also be set if **USER** is set.

  The $user specified must have Log on as a batch job permission, and the source user (the user calling Vortex <EXEC>, typically I_USR if from web server) must have Act as part of the operating system, Increase quotas, and Replace a process level token permissions set. These are requirements of the Windows operating system. **Note: before changing these permissions, discuss the security implications with your Windows administrator.**

  Lack of the first permission may result in Win32 error 1385 (ERROR_LOGON_TYPE_NOT_GRANTED: “Logon failure: the user has not been granted the requested logon type at this computer”), whereas lack of the other permissions may result in Win32 error 1314 (ERROR_PRIVILEGE_NOT_HELD: “A required privilege is not held by the client”).

  If these permissions cannot be set, the FASTLOGON flag may be given, which avoids the need for the permissions, but adds other caveats (see below).

- **PASS=$pass**
  Sets the password to use for logging in the **USER**. Added in version 5. Only supported under Windows.

- **FASTLOGON**
  If this flag is set, a faster logon method that avoids the Windows LogonUser() call is used when executing programs as another user (i.e. **USER** set). Setting this flag avoids the need to set Logon Batch Windows perms for **USER**, and Run as System and Replace OS Token for the Vortex user. However, the <EXEC> process will pop up a new command-prompt window, its output (stdout) may not be available, and the parent process of the Vortex script (e.g. the web server) may wait for the grandchild <EXEC> process indefinitely. Added in version 5. Only supported under certain (recent) Windows versions.

- **QUOTEARGS**
  Under Windows, double-quote the command-line program/arguments that contain spaces and do not already contain double-quotes; under Unix and other operating systems, do nothing. Setting **QUOTEARGS** allows the same arguments to be passed to <EXEC> whether running under Unix or Windows, without having to explicitly quote potentially space-containing arguments under one OS but not others. (Note that it is still up to the individual <EXEC>ed program to properly parse quoted arguments.)

  If neither QUOTEARGS nor NOQUOTEARGS is specified, the flag defaults to the texis.ini setting [Texis] Exec Quote Args (p. 579). If that is not set, the flag defaults to on. Added in version 6.

- **NOQUOTEARGS**
  Do not double-quote program/arguments.

13 Prior to Windows 2000, the equivalent privileges were Logon Batch, Run as System, none, and Replace OS Token, respectively.
CHAPTER 1. TEXIS WEB SCRIPT

- **FLAGS=$flags**
  Set the flags given in the space/comma-separated list $flags. Any of the values bin, bkgnd, envclr, fastlogon, msg, nobr, nomsq, noquoteargs, quoteargs may be given. This allows flags to be set dynamically. Added in version 6.

- **FROMFILE=$file**
  Sets the file to read input from. If empty, input is redirected from /dev/null or NUL. If “-” (a single dash), input comes from the Vortex script’s standard input (aka stdin). Otherwise input comes from the file specified. If FROMFILE is not specified at all, input comes from the output of the enclosed Vortex statements. Added in version 5.

- **STDERRTOFILE=$file**
  Sets the file to redirect standard error (aka stderr) to. If empty, standard error is redirected to /dev/null or NUL. If “-” (a single dash), standard error is redirected to the Vortex script’s standard error. Otherwise standard error goes to the specified file. If STDERRTOFILE is not specified, standard error is returned in the Vortex variable $ret.stderr for foreground programs, otherwise (BKGND) it goes to /dev/null or NUL. Added in version 5; in previous versions standard error went to the Vortex script’s standard error (Unix) or the same place as standard output (Windows).

- **SKIPONFAIL**
  If the program cannot be executed, skip the inner block code. See same option in <WRITE> for details. Defaults to off. Added in version 7; in previous versions it was effectively always on. Note that setting SKIPONFAIL will not skip the inner block code if the program can be run but exits non-zero, as program exit may happen some time after the inner block is entered.

- **TIMEOUT=$n**
  Sets a timeout for the program: if execution exceeds $n seconds, the program is terminated. -1 specifies no timeout, which is the default if unspecified or empty. Added in version 5.

- **CHDIR=$dir**
  Sets the working directory for the program to $dir (without affecting the Vortex current working directory). The default is the same as the Vortex current working directory. Added in version 5. More efficient and modular that using <syscp chdir>.

- **ENVSET**
  Flag that indicates the “option assignments” that follow are actually environment variable assignments to be added for the program, without altering the current Vortex environment. E.g. ENVSET LD_LIBRARY_PATH=/usr/local/bin would add the LD_LIBRARY_PATH environment variable set to /usr/local/bin for the program. Environment variable assignments are taken until the next non-assignment option. Both names and values may be Vortex variables; multi-value names are assigned in parallel with values, with the shorter list’s last value extended to match. An unset Vortex variable used for a value causes the named environment variable(s) to be deleted instead. More efficient and modular than using <syscp setenv>. Added in version 5.

- **ENVCLR**
  Flag that indicates the parent (Vortex) environment is not to be inherited, which is the default. Useful for executing a program in a known fixed environment, since Vortex’s environment may vary (e.g. CGI vs. command line).
• MSG
  Flag that indicates error putmsgs are to be distributed, i.e. printed/logged/captured as indicated by `<vxcp putmsg>` (p. 284). This is the default behavior. If neither MSG nor NOMSG are specified, message distribution is according to `<vxcp execmsg>` (p. 287). Added in version 5. Useful for enabling messages just for one `<EXEC>` when the default is set otherwise via `<vxcp execmsg>`. Note that in version 5, all `<EXEC>` putmsgs are also returned in $ret.err, regardless of MSG/NOMSG/<vxcp execmsg>.

• NOMSG
  Flag that indicates error putmsgs are not to be distributed via the current putmsg mechanisms. If neither MSG nor NOMSG are specified, message distribution is according to `<vxcp execmsg>` (p. 287). Added in version 5. Note that in version 5, all `<EXEC>` putmsgs are also returned in $ret.err, regardless of MSG/NOMSG/<vxcp execmsg>. Setting NOMSG is an easy way to suppress putmsgs (e.g. for exit codes or missing program) for just one `<EXEC>`, without having to set and clear `<vxcp execmsg>` or `<vxcp putmsg>`.

• MAXSTDOUT=$n
  Sets the maximum number of bytes of standard output read from the program; the default if empty or unspecified is no limit, Useful to avoid consuming large amounts of memory for $ret if a program unexpectedly returns lots of output. Added in version 5.

• MAXSTDERR=$n
  Sets the maximum number of bytes of standard error read from the program; the default if empty or unspecified is no limit, Useful to avoid consuming large amounts of memory for $ret if a program unexpectedly returns lots of error output. Added in version 5.

• APPEND
  Flag that indicates the immediately preceding TOFILE or STDERRTOFILE specification should append to the file (if using a file) instead of truncating it (the default). Note that in Windows versions a seek to EOF only occurs at file open, not atomically at each write() as in Unix. This means that two programs APPENDING to the same file over the same time period may erase each other’s output under Windows, instead of interleaving as in Unix. Added in version 5.

• -- (two dashes)
  Flag that indicates the end of options, and that the program name and arguments follow. Useful to avoid confusion if a program has the same name as an option. Recommended for all `<EXEC>` calls to avoid confusion if future undetermined options are added to `<EXEC>` . Added in version 5.

DIAGNOSTICS

EXEC returns the standard output of the program (unless BKGDND or TOFILE is given). If the NOBR or BIN flags are given, the output is one value; otherwise the output is split into one value per line.

EXAMPLE
This example runs a GIF to JPEG translator on the file /tmp/my.gif. The translator uses standard in and standard out, and the resulting JPEG is saved to a file:

```plaintext
<EXEC NOBR FROMFILE="/tmp/my.gif" --
    /usr/local/bin/gif2jpg>
</EXEC>
<write "/tmp/my.jpg"><fmt "%s" $ret></write>
```

CAVEATS

The EXEC statement was added Aug. 29 1996. The BIN and NOBR flags were added Feb. 4 1997. BKGND appeared Apr. 30 1997. Many other options appeared in version 5.

If TOFILE is ”-” (single dash) for stdout, the program’s output may appear out-of-sequence with respect to Vortex’s output and will not be <CAPTURE>-able. This may change in a future version.

SEE ALSO

CAPTURE
1.2. VORTEX STATEMENTS

1.2.14 CAPTURE – capture output

SYNOPSIS

<CAPTURE [flags]>
  ... Vortex code ...
</CAPTURE>

DESCRIPTION

The CAPTURE command redirects (captures) the standard output of the Vortex code within its block, and returns it in $ret at the close of the block. It is useful for saving the output of an entire section of script, rather than laboriously appending many values together. The following flags may be used:

- BIN
  The return value $ret is considered binary (e.g. varbyte). By default the return value is text (varchar).

- OUTPUT
  Also print the output normally to stdout. This is useful if the output is to be displayed anyway, especially if the enclosed code may execute slowly: the output can be displayed as generated while still being captured, without waiting for the entire block to finish.

DIAGNOSTICS

CAPTURE returns the standard output of the Vortex code within the block, as one value.
EXAMPLE

This script uses CAPTURE in a simple page-caching scheme. Common queries are cached in the cache table. If a query is not present already, it is executed and cached:

<!-- Check the cache first: -->
<SQL "select Page from cache where Query = $query">
  <send $Page>
  <RETURN>
</SQL>
<!-- Query wasn’t in the cache; execute and save it: -->
<CAPTURE OUTPUT>
  The following books match your query: <P>
  <SQL "select Title from books where Subject like $query">
    <A HREF="url/details.html">$Title</A>
  </SQL>
</CAPTURE>
<!-- Add it to the cache: -->
<SQL NOVARS "insert into cache values($query, $ret)">
</SQL>

CAVEATS

The CAPTURE statement was added in version 2.1.896900000 19980603.

The output of a CAPTURE block with the OUTPUT flag can still be captured by an enclosing CAPTURE, EXEC, etc. block.

SEE ALSO

EXEC
1.2.15 TIMPORT – general purpose data import

SYNOPSIS

<TIMPORT [options] $schema [FROMFILE] $data|$file>
... statements ...
</TIMPORT>

DESCRIPTION

The TIMPORT statement (Texis IMPORT) imports text data into Vortex variables. The $schema argument is a schema (template), which defines what fields are in the source data and what format they are in.

The $data parameter is the text buffer containing raw data to be imported. If the FROMFILE flag is given, it is considered a filename instead, and that file is read for the import data.

Each value of the buffer or filename variable is imported in sequence. The returned fields are assigned to Vortex variables of the same name, one field per value. Each result row returned by TIMPORT adds another set of values to the variables, and the statements inside the TIMPORT block are executed. (I.e. a loop occurs for every row imported, not just for every buffer/filename given.)

Unlike the command-line version of TIMPORT, in Vortex the imported fields are not inserted into a table. Instead they are returned as variables for further processing by the Vortex script. This gives the programmer complete control over what rows go where and how. For full details on the syntax of schemas and the usage of TIMPORT in general, see the Texis manual on TIMPORT. Note that the following TIMPORT keywords do not apply in Vortex, because no table is used: host, port, user, group, pass, createtable, database, droptable, table.

The variables returned by TIMPORT behave like SQL variables (p. 27) in that they are appended in a list, yet only the current value returned is visible inside the loop. SKIP and MAX behave similarly, giving the number of initial values to skip, and the maximum number of values to return.

The flags NOVARS, OKVARS and ROW can also be used; as in a SQL statement, NOVARS causes no variables to actually be returned, while ROW causes the variables to be returned as single values, instead of being appended in a loop.

The $loop and $next variables are set as in SQL, i.e. the current loop iteration count and next-iteration count. $indexcount is not set, however, nor is $null significant.

A BREAK statement may be given inside the loop to exit prematurely.

EXAMPLE
The following example imports fields from a Unix-style mailbox into the variables $From, $Subject, $Date and lists each message:

```xml
<$schema = "
# take multiple records from a single file:
multiple
datefmt x, dd mmm yyyy HH:MM:SS
# name type tag default_val
field From varchar />>\RFrom\x20\P=[^\space]+ UNKNOWN
field Subject varchar Subject UNKNOWN
field Date date Date
field Body varchar />>[^\alnum\-_]+:\x20[^\n]+\n\nP=!=\nFrom\x20+
"/>
<TIMPORT ROW $schema FROMFILE /tmp/mbox>
  From: $From <BR>
  Subject: $Subject <BR>
  Date: $Date <BR>
  <P> <PRE>
  $Body
  </PRE>
  <HR>
</TIMPORT>

CAVEATS

The TIMPORT statement was added in version 2.1.872500000 19970825.

Column variables returned by TIMPORT are cleared first before the loop starts, i.e. previous values are lost. However, if no rows are returned by the command, then the variables are not cleared.

The rows skipped by SKIP apply after any row(s) skipped by TIMPORT schema keywords. Thus, if columnar data (format col or csv) is imported without the keepfirst keyword, the first row is still skipped, even if SKIP is 0.


SEE ALSO

SQL, LOOP, BREAK
1.2. VORTEX STATEMENTS

1.2.16 BREAK – exit loop

SYNOPSIS

<BREAK>

DESCRIPTION

The BREAK statement exits the innermost enclosing looping statement, such as WHILE, SQL, LOOP, etc. It is useful for exiting a long loop quickly when the remaining iterations aren’t needed.

EXAMPLE

<TIMPORT ROW $schema FROMFILE /tmp/mbox>
  <!-- stop before today’s messages: -->
  <IF $Date gt "start of today">
    <BREAK>
  </IF>
  ... process current message ...
</TIMPORT>

CAVEATS

The BREAK statement was added in version 1.0.855300000 19970206.

Note that unlike in C, BREAK does not exit a SWITCH statement.

SEE ALSO

CONTINUE
CHAPTER 1. TEXIS WEB SCRIPT

1.2.17 CONTINUE – continue loop

SYNOPSIS

<CONTINUE>

DESCRIPTION

The CONTINUE statement continues the innermost enclosing looping statement, such as WHILE, SQL, LOOP, etc., at the start of its next iteration. It is useful for skipping a loop iteration near the top, without a long <IF> block over the remaining loop code.

EXAMPLE

<TIMPORT ROW $schema FROMFILE /tmp/mbox>
   <!-- skip messages from "MAILER-DAEMON": -->
   <rex "MAILER-DAEMON" $From>
   <IF $ret neq ">
      <CONTINUE>
   </IF>
   ... process current message ...
</TIMPORT>

CAVEATS

The CONTINUE statement was added in version 4.0.999050000 20010828.

SEE ALSO

BREAK
1.2. VORTEX STATEMENTS

1.2.18 RETURN – exit current function

SYNOPSIS

<RETURN [$value]>

DESCRIPTION

The RETURN statement ends the current function, optionally setting the value of $ret to its given argument. It can be used to exit a function early when some condition or error is met.

DIAGNOSTICS

RETURN sets a return value if an argument is given.

EXAMPLE

<A NAME=update user>
  <SQL MAX=1 "select FullName from passwd where User = $user">
  </SQL>
  <IF $loop eq 0> <!-- no such $user: quit now -->
    <RETURN>
  </IF>
  ... continue to update the user ...
</A>

CAVEATS

The RETURN statement was added in version 2.1.882300000 19971216.

SEE ALSO

BREAK, exit
1.2.19 VERB – raw output

SYNOPSIS

<VERB [NOESC]>
... raw text to be output ...
</VERB>

DESCRIPTION

The verb element causes raw output without interpretation by Vortex. Any Vortex statements, directives, variables, etc. are ignored, including the usual stripping of leading whitespace, up to a closing </VERB> tag.

HTML-sensitive characters, such as <, and >, will be HTML-escaped on output, so raw HTML can be placed in the element. If the optional NOESC attribute is given, this HTML escapement is not done.

DIAGNOSTICS

VERB has no effect on $ret.

EXAMPLE

<VERB>
  This text is uninterpreted:
  <IF $x lt $y>
    yes
  </IF>
  <test>
</VERB>

produces the following output (note leading spaces):

  This text is uninterpreted:
  &lt;IF $x lt $y&gt;
    yes
  &lt;/IF&gt;
  &lt;test&gt;

CAVEATS
The **VERB** statement was added Aug. 28 1996.

Note that a **VERB** element does *not* imply an HTML `<LISTING>` tag, i.e. fixed-font, literal text interpretation by the Web browser. It only prevents interpretation by Vortex; a `<LISTING>` element must be explicitly given if desired.

**SEE ALSO**

`send`
1.3 Vortex Directives
1.3. VORTEX DIRECTIVES

1.3.1 TIMEOUT – set timeout

SYNOPSIS

<TIMEOUT = n [ / ]>

... text/HTML to print on timeout ...

[</TIMEOUT>]

DESCRIPTION

The TIMEOUT directive sets a timeout (in seconds) for the execution of the Vortex script. If script execution takes longer than \( n \) seconds, the enclosed text or HTML is printed and the script exits. Command-line option -t overrides this directive, and <vxcp timeout> overrides -t. The text to print can be overridden at runtime (in version 7 and later) with <vxcp timeout=text>. The default timeout is 30 seconds.

As it is a directive, TIMEOUT must appear before the first function in the script.

EXAMPLE

<TIMEOUT = 10>

<H4>Time Exceeded</H4>

Your query exceeded the time limit.

</TIMEOUT>

CAVEATS

The TIMEOUT directive was added Sep. 4 1996.

The \( n \) argument must be an integer literal, since TIMEOUT is evaluated at compile time. No Vortex commands may appear inside the TIMEOUT element.

If a timeout of -1 is given, the timeout is disabled (no time limit). This is not recommended in a web environment: multiple invocations of a time-consuming script could potentially pile up and load the server.

The execution time of a script may also be affected by a time limit the web server imposes on CGI programs: the server might kill the script before the TIMEOUT directive does. Consult your web server manual.

The script timeout can be overridden at runtime with <vxcp timeout> (p. 284).

Do not confuse the script timeout (<TIMEOUT> directive) with the network timeout (<urlcp timeout>) or JavaScript timeout (<urlcp scripttimeout>). The network timeout should be smaller than the script timeout (<TIMEOUT>) when fetching pages, or the script may end before the fetch error is returned.
SEE ALSO

vxcp
1.3.2  EXPORT – retain variables across invocations

SYNOPSIS

<EXPORT $var [$var ...] [flags]>

DESCRIPTION

The EXPORT directive marks the given variables for “export” to the next invocation of the script. Their values will be retrieved the next time the script is run.

EXPORTed variables are saved when the special variable $url is printed, whose value is the URL to re-invoke the script (see p. 84). Variables EXPORTed with the QUERY flag are saved to $urlq, whose value is a URL-encoded query string (starting with a “?” if non-empty) containing the QUERY variables.

When a user then clicks the $url link, not only is the script invoked as usual, but all variables EXPORTed at the time the $url was printed, are restored in the new invocation. Thus the script can remember a user’s query, login id, etc. across multiple runs.

EXPORT Types

EXPORTed variables are saved in one of three ways:

- State table
  Variables that are given the flag TABLE, or are not currently in a LOOP or non-ROW SQL, are written to the state table (vortex). These are known as state-table variables. Variables which have large values, only change once per invocation, or are used globally are best saved this way. A user name, or “shopping cart” list of items selected so far are typical examples.

- URL state
  Variables that are given the flag URL, or are in a loop, have their current value(s) saved, but encoded to the URL instead. These are known as URL state variables. Variables which have small values, or multiple values per invocation (with each value to go to a different URL) are best saved as URL state variables. A good example is the id field for each result of a first-level query: each id goes to a different URL, for details on that hit. URL state variables also avoid the overhead of reading and writing to the state table, and thus let the script run somewhat faster.

- Query string
  Variables that are given the flag QUERY are saved as a URL-encoded query-string to the special variable $urlq. This can be useful for tracking script usage in web server logs, as the Vortex variables will be in plain view. It can also be used to export variables to another script.

When $url is printed, its value will reflect both the encoded URL state variables (if any), as well as a handle to the state-table variables. Similarly, when $urlq is printed, query-string variables are reflected in it.
The state table is written to only *once* during a given script invocation, to save transaction time. It is written the first time $url$ is printed when at least one state-table variable is set. At that point all state-table variables are saved.

URL state variables, however, are saved to the $url$ value *every* time it is printed (the same is true for $urlq$).

Any EXPORTed variables that are not set when $url/$urlq is printed are not exported at that time.

**Flags**

The following flags may appear after the variable list in the EXPORT directive:

- **TABLE**
  Forces the listed variables to be exported to the state table, even when in a LOOP.

- **URL**
  Forces the listed variables to be exported to the URL state, even when not in a LOOP.

- **QUERY**
  The listed variables will be exported in a URL-encoded query string ($urlq$). Note that the values will be plainly visible, and that type information (e.g. string vs. integer) may be lost on re-invocation, since the variables are just strings. Also, the values will have the lower initialization precedence of query string variables, since they’re not true state variables.

- **USEROK**
  Allows “user” variables – URL query string and content variables, i.e. from an HTML form – to override state values on initialization.

If neither URL, TABLE nor QUERY is given, where the variables are exported depends on whether they’re in a LOOP at the time (see above). The URL, TABLE and QUERY flags are mutually exclusive.

The USEROK flag alters the precedence for the listed variables, making form values have the highest precedence for initialization, instead of state values (the default). This affects which values are used to initialize a variable on startup, when more than one source is present (e.g. a variable was submitted in a form and present in the URL state).

Normally, the state values (URL and TABLE) have precedence over any other source. This is because state values are controlled by the script and are considered (semi-)private, but the form variables are controlled by the (possibly malicious) web user: the script’s values are preferable. Thus any extraneous form variables sent by the browser do not normally affect EXPORTed variables.

However, in some cases it may be desirable to allow form variables to override state values. For example, a user’s query may be EXPORTed to save it across invocations for future searches, but the user should be able to alter it at any time. In this case, the USEROK flag should be set:

```<EXPORT $query USEROK>
...```
In the above example, the user’s $query value is remembered across invocations because it is EXPORTed. However, a new value can be set from the form and will override the saved value on submission, because USEROK is set. Without this flag, the original value of $query would be preserved: the user would not be able to change it.

EXAMPLE

A book search might generate a list of matching books, with a URL for each book. Each URL brings up details on that book, based on its id. In addition, the user’s query, $query, is to be remembered across invocations, in a format easily viewed in server logs. Also, we want to remember the user’s name from their login. Thus, $id, $query and $user are exported:

<EXPORT $id>
<EXPORT $query QUERY>
<EXPORT $user>

Hello, $’user’. These are the books that match your query:
<SQL "select id, Title from books where Title like $query">
  <A HREF=$url/details.html$urlq>$Title</A> <BR>
</SQL>

The variables $id and $user are saved when $url is printed. Since $user is not being looped over in the SQL loop, it is a state-table variable when $url is printed, and is saved once, to the state table. All the URLs generated in the SQL loop will have the same value(s) for $user. This is fine, because $user is unlikely to change during the session.

However, $id is a loop variable, because it is set by the SQL loop: thus it is saved to the URL state instead. Each URL will have the same value for $user, but a single, different value of $id, for the appropriate book. The details function can then select the given book by $id and print more detailed info on it.

The $query variable is flagged QUERY, so it is exported to a query string when $urlq is printed. Though it has a constant value in this SQL loop, were it to change each time it would have a different value in $urlq. Its value is plainly visible in the overall URL, thus enabling easy query tracking via Web server logs. Note that it is appended after the function/MIME part of the URL.
CAVEATS

The QUERY flag was added in version 2.1.898900000 19980626.

EXPORT is a directive, and as such must appear before the first function in the script. Variables are not actually saved, however, until the $url variable is printed.

The $url and $urlq variables must be printed to initiate state retention: referring to them in an assignment or function call will not save state.

The maximum URL length is arbitrarily fixed at 512 characters to help prevent truncation by browsers; if URL state variables have values that are too long when $url is printed, some or all of the variables will not be exported.

Note that state-table variables are only saved once to the state table. URL and query state variables, however, are written to the URL any number of times, every time $url or $urlq is printed.

Variables are exported from the scope of the $url location. This means that a local variable of the same name as an EXPORT will be EXPORTed if it is in scope when $url is printed.

The state table vortex (and the database if needed) is created on the fly at run time if it does not exist, whenever $url is printed or a URL with state variables is accessed. Since old state information is not erased from the table, it is advisable to occasionally clear it with the -W command-line option lest it grow too large.

EXPORTing any data to URL should not be considered secure, as the data is compressed but not encrypted. EXPORTing to QUERY will present data in the clear to users, as this is intended for logging. Any sensitive data such as login info should be saved via another mechanism, e.g. encrypted to a cookie.

The USEROK and QUERY flags should only be given for variables that are expected to be modified by the Web user. The QUERY flag lowers the initialization precedence of the variables, since they are not true state variables but are part of a query string. Variables EXPORTed this way may also lose type information (or suffer truncation if binary) because they are stored as strings. EXPORTing as QUERY may also increase Web server log size.
1.3.3 PUTMSG – set error message actions

SYNOPSIS

<PUTMSG call|log|print|all|override on|off|exceptiononly>

DESCRIPTION

The PUTMSG directive sets the disposition of error messages. When a script generates errors, they are logged to vortex.log and printed out by default. If a <putmsg> function is defined, then logging and printing are turned off, and the function is called instead (see p. 581 for details).

By setting the appropriate action (call, log, print, or all for all three) in a <PUTMSG> directive, a different set of call/log/print actions can be set for the script as desired. A typical use for this is to turn logging and printing back on when a <putmsg> function is defined, so that errors can be monitored on-the-fly by the script itself, but are still logged for later perusal, without having to log them “by hand” in the <putmsg> function.

As it is a directive, PUTMSG must appear before the first function in the script, and sets the error message actions for the entire script. The <vxcp> function (p. 284) can be called to override these settings at run-time, via its putmsg setting.

The override action has no equivalent in <vxcp putmsg>. It makes any other <PUTMSG> directives override <vxcp putmsg>, instead of the other way around. The use for this potentially confusing setting is debugging: a script with a complex set of <vxcp putmsg> calls can have them all temporarily disabled by adding <PUTMSG override on> and <PUTMSG log on> at the top temporarily, without having to edit the rest of the script. In version 7.03.1430345000 20150429 and later, override also overrides <urlcp putmsg pass> (p. 229), i.e. it is always on. The override action value is Boolean only, i.e. on or off only.

The log and print actions may be set to a third state, exceptiononly, in Texit version (or compatibilityversion)7 and later. This is their default value when a <putmsg> script function is defined. When exceptiononly is set, the given action is only enabled if a Vortex putmsg exception is occurring, i.e. a situation where the <putmsg> script function cannot be called because the script is terminating. Examples include the script <timeout> firing, an ABEND or stack overflow. For all other normal situations – where <putmsg> can be called – the action is disabled. Thus, exceptiononly allows printing and/or logging to be turned off for messages than <putmsg> can handle, yet still allows them to be seen and not lost if the script terminates abnormally. The exceptiononly value cannot be set for the call action.

EXAMPLE

<PUTMSG log on>

<A NAME=putmsg PRIVATE>
<$GotErr = y>
</A>

<A NAME=main>
<EXEC /some/program></EXEC>
<IF $GotErr eq "y">
   <B>An error occurred: check the logs</B>
</IF>
</A>

In this script, an error from the <EXEC> is still logged, as well as being detected by the script via <putmsg>.

CAVEATS

The PUTMSG directive was added in version 3.0.947100000 20000105.

SEE ALSO

vxcp, Error Messages section (p. 581), ErrorScript ErrorFile settings in texis.ini (p. 576)
1.3. VORTEX DIRECTIVES

1.3.4 ENTRYFUNC, EXITFUNC – set entry/exit functions

SYNOPSIS

<ENTRYFUNC = function>
<EXITFUNC = function>

DESCRIPTION

The ENTRYFUNC directive sets an additional function to call first upon entry to the script, before the start function (which is specified in the URL or $cmd). By default, no entry function is called, i.e. execution starts with the start function. The EXITFUNC directive sets an additional function to call when the script ends normally (i.e. not via <exit>). By default no exit function is called.

Since execution of both functions is controlled by the script and not the user, private functions may be specified. As they are directives, both ENTRYFUNC and EXITFUNC must appear before the first function in the script.

An entry function can be useful for setting global variables, checking access permissions, or printing the look and feel HTML for an application. Since it is guaranteed to be called first – before the start function or main – it does not need to be explicitly called by every public function in the script, and will be called for any future public functions added. Once the entry function returns, execution resumes at the start function (or main) as normal. Thus, the entry function can concentrate on tasks that are the same for every invocation, while the specific start function(s) can deal with function-specific tasks (e.g. displaying vs. editing). An entry function has advantages over EntryScript (p. 593) in that it can be used by any script, including command-line and non-Texis-Web-Server scripts, and it can set variables and call functions within the script.

Modules can also have entry/exit functions; the entry function for a module is called before the entry function for the parent file that <uses> it, so that the parent entry function can count on its modules already being initialized. (However, there is no guaranteed call order for multiple modules’ entry functions in a single <uses>, nor is the entry function for a module that is used by multiple parents guaranteed to be called before all of its parents’.)

An exit function is useful for logging transactions or printing the bottom part of look at feel HTML. It is called when the script ends normally, i.e. not via exit.

EXAMPLE

<ENTRYFUNC = checkperms>
<EXITFUNC = logtransaction>

<A NAME=checkperms PRIVATE>

14Except for ErrorScript invoked during errors, where putmsg may be invoked first.
<if "127.0.0.1" neq $REMOTE_ADDR>
  You are not authorized to use this application. <exit>
</if>

<HTML><HEAD><TITLE>My Application</TITLE></HEAD>
<BODY BGCOLOR=white>
</A>

<A NAME=logtransaction PRIVATE>
<SQL "insert into log(ip, query)
values($REMOTE_ADDR, $QUERY_STRING)">
</SQL>
</BODY>
</HTML>

<A NAME=main PUBLIC>
Hello.
</A>

<A NAME=edit PUBLIC>
... edit form ...
</A>

Every run of the script – whether to main or edit or some future function – will first execute checkperms to check permissions and print opening HTML. At the end (if exit is not called), logtransaction will be called to log the transaction.

CAVEATS

The ENTRYFUNC and EXITFUNC directives were added in version 5.01.1098192447 20041019.

SEE ALSO

EntryScript (p. 593), ExitScript (p. 593),
1.3.5 USES – use a module

SYNOPSIS

<USES [DATE=date] module[=file] [module[=file]]>

DESCRIPTION

The USES directive includes various other modules (scripts) for use in the current script. Each module named in the directive is included when compiling the script, as well as any modules that those modules include. By default the latest revision of each module is used; if the DATE option is given then the latest revision on or before that date is used. This enables a script to “freeze” its usage in time and ignore later revisions, if for example the calling syntax for some functions in the module has changed, but the script hasn’t updated its usage.

An optional filename may appear after the module name with an equal sign. This indicates that the module is to be read from the named file instead of the library. It overrides any inclusion of that module by other modules. If not absolute, file paths are interpreted relative to the including script’s path. By including a module directly from a file, a new revision can be tested independently without repetitive checkins on every edit, nor disturbing other live code that uses the module. **Note:** Once the revision is considered stable, it should be checked in to the library to speed up the script, as modules included from files must be checked for changes on every script invocation, unlike library modules.

Module functions are visible to the script according to their scope; see the discussion of function scope on p. 12 for details. Most module functions that are intended for use by other scripts should be declared EXPORT, so that other scripts can see them, but the end user cannot call them and disrupt script execution.

When any listed module is later changed, a script that <USES> the module – directly or indirectly – will be flagged for re-compilation on invocation. This ensures that scripts are kept up-to-date with respect to the modules they use, so that manual re-compilation is not needed whenever a module changes.

As it is a directive, USES must appear before the first function in the script.

For more information, see section 1.8 (p. 558) on library modules.

EXAMPLE

<SCRIPT LANGUAGE=vortex>

<USES security>

<A NAME=main>
  <verifyuser>
  <IF $ret neq "ok">

15 This feature appeared in version 3.01.975500000 20001129.
In the above example, a module security is included, which contains the function `<verifyuser>` written by the programmer. This function is called in `<main>` to verify the user’s identity. Since `<verifyuser>` is in a module, any script can use it.

CAVEATS

The `USES` directive was added in version 2.6.936300000 19990902.

Modules included as files instead of from the library increase the script’s run time slightly, since these files must be checked for modification on every run. Once testing of a file module is completed, it is advisable to check it in to the library and remove the “=file” part of the clause to avoid this check.

SEE ALSO

Vortex Library Modules
1.3. VORTEX DIRECTIVES

1.3.6 SCHEDULE – schedule periodic execution of script

SYNOPSIS

<SCHEDULE [options] WHEN=schedule [---] [var=val] [...]>

DESCRIPTION

The SCHEDULE directive causes the script to be periodically executed, according to the given WHEN schedule. The WHEN schedule is a string with one of the following syntaxes:

- minutes
- [EVERY] [n] DAY[S]|DAILY timeofday [minutes]
- [EVERY] [n] WEEK[S|LY] [ON] weekday timeofday [minutes]
- [EVERY] [n] MONTH[S|LY] [ON] [THE] FIRST|SECOND|THIRD|FOURTH|n [TO LAST] weekday|DAY [OF THE MONTH] timeofday [minutes]

with these additional definitions:

minutes = [EVERY] [n] MINUTE[S] [ [REPEAT] n [TIME[S]]
timeofday = AT [h]h[:][m]m[A|P[M]]
weekday = SUN|MON|TUE|WED|THU|FRI|SAT |
          SUNDAY|MONDAY|TUESDAY|WEDNESDAY|THURSDAY|FRIDAY|SATURDAY

Alternatively, the WHEN schedule may be a string with the vCalendar\textsuperscript{16} 1.0 scheduling syntax (available at http://www.imc.org/pdi/pdiproddev.html).

SCHEDULE may also be called as a function; it returns 0 on error, 1 if ok, and 2 if nothing done (scheduler disable via texis.ini). The primary function usage is to unschedule a script (with WHEN set to “off”) or all scripts (“all-off”).

In addition to the WHEN schedule, the following options may appear in the SCHEDULE directive or function:

- START=date
  The starting date for the schedule, as a Texis-parseable date/time. This defaults to now. The first run of the schedule will be the next valid WHEN event on or after START. It is recommended that a START date be given if the WHEN argument is of the EVERY [n] MINUTES syntax, if the script is to start at a fixed wall-clock time (e.g. every hour, but on the hour).

- URLEND=/func.txt
  Specifies the function and MIME-type file extension (p. 8) to be appended to the script name when run. This allows a function other than <main> to be the starting point for the script run.

\textsuperscript{16}vCalendar is a trademark of Apple Computer, Inc., AT&T Corp., International Business Machines Corp., and Siemens.
• **COMMENT=text**
  Specifies a short text comment to be associated with this schedule. This is viewable when the `-LS` command-line option is used (p. 563), and can be used to denote the purpose of each scheduled event.

• **SCRIPT=file**
  Script to schedule/unschedule. Only valid if called as function; directive script is always current script. Default is main script (non-module).

• **FLAGS=flags**
  Set space-separated list of flags. May include:
  
  - **mutex**
    Run jobs mutually-exclusive: do not start a scheduled run of the script if a previous run of the same script same schedule is still active. Scripts should still run their own mutex check however, as this only prevents scheduled collisions, not web or command-line runs from colliding.

  - **nomutex**
    Do not mutually-exclude runs. Default in version 7 and earlier.

  **FLAGS** was added in version 7.02.1409016000 20140825. Note that previous versions will take this option as a variable assignment.

• **--**
  Indicates end of options, and that all arguments that follow (if any) are variable assignments to pass to the script. Ensures that any var assignments will not be confused with options. Added in version 7.02.1409016000 20140825.

After all options, a list of variable assignments may be given. These will be passed to the script when it is run, allowing user-defined configuration options to be set.

Multiple `<SCHEDULE>` tags are permitted, e.g. to run different parts of the script at different times. To unschedule a script, remove the `<SCHEDULE>` tag and re-compile or re-run the script, use `-sched all-off, or -wipesched (p. 563).`

For more information on scheduling, see the Scheduling section, p. 562.

**EXAMPLE**

```
<SCHEDULE WHEN="daily at 10pm">
</SCHEDULE>

<SCHEDULE WHEN="every month on the first Sunday at 4am"
      URLEND=/monthly.txt>
</SCHEDULE>
```

The first example schedules the script to run the `<main>` function every day at 10pm local time. The second example runs the `<monthly>` function on the first Sunday of every month, at 4am.
CAVEATS

The <SCHEDULE> tag was added in version 3.01.985400000 20010323. The function syntax was added in version 3.01.988150000 20010424.

The script must be compiled, either explicitly via the –C option or implicitly by running the source, for the <SCHEDULE> tag to take effect, since Vortex must be made aware of it.

The Texit Monitor (monitor) is responsible for starting scheduled scripts, so it must be running for SCHEDULE to work. Running texit, tsq1 or vhttpd will start the Texit Monitor; alternatively it may be automatically run by the OS at machine boot (under Windows the install program for Texit arranges this). Make sure it is run as the Texit user, not root.

SEE ALSO

Vortex Scheduling
1.3.7 COOKIES – control cookie import

SYNOPSIS

<COOKIES urldecode|asis>

DESCRIPTION

The COOKIES directive controls how HTTP cookies are imported into Vortex variables upon script start.

The default setting of urldecode will URL-decode cookie values when initializing Vortex variables, i.e. assume they are URL-encoded. The setting of asis will not decode values, i.e. leave them as-is. This setting can be useful to import cookies set by another application that does not URL-encode the values. Note that cookie names are never decoded.

The COOKIES directive overrides the system-wide default set by the [Texis] Cookies setting of texis.ini (p. 579).

EXAMPLE

<COOKIES asis>

<A NAME=main>
  <if "" neq $ASPSESSIONID>
    ... check ASP session info ...
  </if>
</A>

CAVEATS

The <COOKIES> directive was added in version 5.01.1121884211 20050720. It defaulted to urldecode in previous versions.

SEE ALSO

[Texis] Cookies setting of texis.ini
1.3.8 STACK – set stack limit

SYNOPSIS

<STACK=n>

DESCRIPTION

The STACK directive sets the maximum stack depth permitted for the script. The stack depth increases by one for every function call and upon entering most block or looping statements. Exceeding the stack limit will generate a “Stack overflow” message and terminate the script. The default limit is 250; -1 sets an infinite (no) limit. Increasing the stack limit is needed only by scripts that make heavy use of recursion and will intentionally exceed the default limit. The usual cause of a stack overflow message is unintentional infinite recursion: a function calling itself (directly or through other functions) in a loop without returning.

As it is a directive, <STACK> must appear before the first function in the script. The stack limit can also be modified at run time with the vxcp function (p. 284), which overrides this directive.

EXAMPLE

<STACK=500>

<A NAME=bigtreewalk>
  ...
</A>

CAVEATS

The <STACK> directive was added in version 3.0.947100000 20000105.

Setting a large or infinite stack limit can allow a script to consume inordinate amounts of memory and CPU, possibly bringing other processes or even the entire operating system down.

SEE ALSO

vxcp
1.3.9 TRACESQL – trace SQL calls

SYNOPSIS

<TRACESQL on|off|N>

DESCRIPTION

The TRACESQL directive sets an integer value that controls the tracing of SQL calls made by a script. (Setting it to "on" or "off" is the equivalent of 1 or 0, respectively.) The lowest 4 bits are interpreted as a value from 0 to 15 that control SQL statement printing, with each value printing everything at lower values too:

- 0: no tracing
- 1: prints a message for every SQL statement, containing the full statement with its parameters
- 2: prints the entirety of parameters instead of just the first few characters
- 3: parameters are single-quote-escaped for cut-and-paste to a tsql command line for testing (not recommended for production use)
- 4: spaces/comments/newlines are not stripped

The remaining higher bits are taken as flags to control other SQL tracing messages. Thus the following values can be bitwise ORed into the value:

- 0x00010: SQL cache debug messages will be printed
- 0x00020: print the database during cache trace messages
- 0x00040: print Texis API calls
- 0x00080: print process ID as well
- 0x00100: do not replace non-ASCII bytes in parameters with “." 
- 0x00200: print compiled SQL expression cache messages
- 0x00400: print compiled SQL expression at compile or open
- 0x00800: print compiled SQL expression before evaluation
- 0x01000: print compiled SQL expression after evaluation
- 0x02000: print SQL expression if non- compilable
1.3. VORTEX DIRECTIVES

- 0x04000: print SQL expression source before parameters parsed
- 0x08000: print SQL expression source after parameters parsed
- 0x10000: print SQL expression value after evaluation

As it is a directive, <TRACESQL> must appear before the first function in the script. SQL tracing can also be enabled at run time with the sqlcp function (p. 126), which overrides this directive, and on the command line with the -tracesql option. Note that tracing functionality and values may change in future releases, as this is a debugging directive. Values 0x04000, 0x08000 and 0x10000 were added in version 7.06.1517417000 20180131.

EXAMPLE

<TRACESQL on>

<a name=main>
  <if $del eq "y">
    <sql "delete from books where id = $id"></sql>
  <else>
    <sql "update books set Text=$txt where id = $id"></sql>
  </if>
</a>

CAVEATS

The <TRACESQL> directive was added in version 3.0.947100000 20000105.

Setting <TRACESQL> to 2 or higher can produce large messages for statements like the insertion of text fields.

SEE ALSO

sqlcp, -tracesql command line option
1.3.10 SQLCACHE – control SQL handle caching

SYNOPSIS

<SQLCACHE = n>

DESCRIPTION

The SQLCACHE directive sets how many idle Texis (SQL) handles to cache for SQL statements, SQL expressions and state-table exports. The default is 2, which is adequate for most scripts.

The cache is automatically increased when needed, e.g. if nested SQL statements exceed the cache size. Generally the only time the cache should be explicitly increased with SQLCACHE is to optimize scripts that repeatedly re-use a lot of the same SQL-expression assignment or SQL statements in a loop. In such cases setting the cache to the number of unique re-used statements may speed up the process. The larger the cache, the more file handles and memory used, so it is best not to make the cache too large.

EXAMPLE

<SQLCACHE = 5>

CAVEATS

SQLCACHE is a directive and as such must appear before any functions. It first appeared in version 2.1.873200000 19970902.

If the SQLCACHE value is set too high, the process may run out of file handles, causing errors.

SEE ALSO

SQL, sqlcp
1.3. VORTEX DIRECTIVES

1.3.11 TRAP – trap signals

SYNOPSIS

<TRAP [CONNRESET=on|off|N] [[SIGNALS=]on|off|N]>

DESCRIPTION

The TRAP directive sets whether to trap signals or not; by default signals are trapped. This directive is for diagnostic purposes and is not typically used by production scripts. The SIGNALS value is a bit-wise ORed integer of the following values:

- 0x0001: Catch non-ABEND signals (e.g. SIGTERM)
- 0x0002: Catch ABEND signals (e.g. SIGSEGV)
- 0x0004: Try to dump core after ABEND via NULL dereference
- 0x0008: Try to dump core after ABEND via signal return
- 0x0010: Print registers at ABEND
- 0x0020: Print 1KB of stack at ABEND
- 0x0040: Print 16KB of stack at ABEND
- 0x0080: Print location details (if known) at ABEND
- 0x0100: Ignore SIGHUP signals (Unix)
- 0x0200: Treat timeout as ABEND, i.e. try to dump core if 0x0004 set
- 0x0400: Print command line of signalling PID if known
- 0x0800: Same info as 0x0400, but also for signalling PPID. If 0x0400 set too, print info for signaller’s entire ancestry.
- 0x1000: Print a backtrace for ABEND signals

The default value “on” is the same as 0x1493; “off” is 0.

The value 0x0080 was added and became part of the default in version 5.00.1088099065 20040624. The value 0x0100 was added in version 5.01.1103822414 20041223. The value 0x0200 was added in version 7.02.1405112000 20140711. The values 0x0400 and 0x0800 were added (and 0x0400 added to default) in version 7.05.1449114000 20151202. The 0x1000 flag was added (and added to default) in version 7.06.1468000000 20160708. The 0x0010 flag was added to the default in version 7.06.1512512000 20171205.

The CONNRESET option was added in version 3.01.989300000 20010507, and controls whether to trap connection-reset on stdout, i.e. when the remote browser prematurely terminates the connection (user hits stop button). Its bit flags are defined as follows:
• 0x0001: Trap SIGIO SO_ERROR recognized error
• 0x0002: Trap SIGIO MSG_PEEK recognized error
• 0x0004: Trap SIGIO MSG_PEEK no data
• 0x0008: Log truncated standard-out data if possible
• 0x0010: Debug print on every SIGIO
• 0x0020: Continue; do not exit if connection-reset detected

The default value “on” is the same as 7; “off” is 0.

The TRAP directive can be overridden at run time by the command-line option -x (for the SIGNALS value only; p. 567), which in turn can be overridden by the trap setting of the vxcp function (p. 284).

EXAMPLE

<TRAP off>

<A NAME=main>
  ... script code to analyze ...
</A>

CAVEATS

The <TRAP> directive was added in version 3.0.947100000 20000105.

Turning off signals can cause aberrant behavior, particularly with <EXEC> and <TIMEOUT>. Not all SIGNALS flags are supported on all platforms; in particular the printing flags. The bit flags may change in future versions. The values are documented as of version 4.03.

SEE ALSO

vxcp
1.3.12 ADDTRAILINGSLASH – add trailing slash to “directory” URLs

SYNOPSIS

<addtrailingslash = yes|no>

DESCRIPTION

The <addtrailingslash> directive controls whether to add a trailing slash (via a redirect) when a

script is accessed via a “directory” URL without a trailing slash. A “directory” URL looks like a directory,

i.e. it contains neither a function/MIME-extension nor user-path information. For example,

“/texis/dir/script” (where the Vortex script is the file “script”) would be redirected to

“/texis/dir/script/”.

Adding a trailing slash to such URLs allows relative links on such a page – such as “search.html” to

enter the <search> function – to be resolved correctly by the browser. Relative links also reduce the

content size of the page, and permit the script to be mirrored or proxied from a different site without having
to rewrite its links. Most web servers do a similar redirect for directories, for the same reasons.

As it is a directive, <addtrailingslash> must appear before the first function in the script. It

overrides the [Texis] Vortex Add Trailing Slash setting in texis.ini, p. 579.

EXAMPLE

<addtrailingslash = yes>

CAVEATS

The <addtrailingslash> directive was added in version 5.01.1227570000 20081124.

SEE ALSO

[Texis] Vortex Add Trailing Slash setting in texis.ini
1.3.13 pragma – inline compiler directive

SYNOPSIS

<!-- pragma [push|pop] name [value] -->

DESCRIPTION

The pragma directive, unlike any other directive, can appear anywhere in the script, including inside a function. Pragmas typically alter the way code is lexically analyzed, so that the pragma can be applied only to certain code sections if desired. Since they appear in comments, earlier Vortex versions that are not aware of a particular pragma will ignore them silently. The pragmas available are:

- **strictcomment on|off**
  Turn on or off strict comment parsing. With strict comments on, comments must start with “<!--”, not just “<!”, thus tags such as “<!DOCTYPE>” are directly printable and do not get interpreted as comments. The default is on. Added in version 3.01.986950000 20010410.

- **nestcomment on|off**
  Turn on or off nesting of comments. With nesting on, comments may be nested. The default is off. Added in version 3.01.986950000 20010410.

- **literalprint on|off**
  Whether to allow literal text and/or variable printing. With literal printing off, printing text or variables by simply placing literal text or $-var references in the code is not allowed and will generate the compile error “Literal/variable printing not allowed when pragma literalprint off”. Whitespace is still allowed but is silently ignored (instead of being output where not part of indentation); comments are also still permitted and ignored.

  Turning off literal printing can be useful in some situations – such as when generating an image – to ensure that all output is being generated by explicit code, e.g. via <fmt>, and no inadvertent whitespace or mistyped code is causing extraneous output, e.g. calls to non-existent functions being interpreted as literal tags.

  The default is on. Added in version 5.01.1223689000 20081010.

- **compilesqlexpressions on|off**
  Whether to compile SQL expressions into the .vtx object file where possible, rather than interpret them on the fly at run-time. SQL expressions are used in parenthetical variable assignments (e.g. <$x = ($y + 5)> and complex <if> statements, amongst other places. Compiling them speeds up script execution, as the original expression then does not need to be re-interpreted by the SQL engine every time the statement is run. The default is the value of the texis.ini setting [Texis] Compile SQL Expressions; if that is unset, the default is on. Added in version 6.01.

In version 7 and later, pragmas can be pushed or popped on a stack. Pushing a pragma sets the value given, but first preserves the current value; popping restores that previous value. Pushes may be nested to any
depth. This allows local sections of code to alter a pragma’s value, while still preserving and restoring the outer code’s value (without needing external “magic” knowledge of what it is):

```
... literalprint could be on or off here (original value)
<!-- pragma push literalprint off -->
<!-- literalprint is off here -->
<!-- pragma pop literalprint -->
literalprint is back to its original value
... 
```

Each pragma has its own independent stack, which starts out empty (causing the pragma’s default value to be in effect). A normal pragma “set” directive (i.e. a pragma with neither push nor pop) acts as a pop (if stack is not empty), followed by a push. It is an error to explicitly pop a pragma when its stack is empty.

Pragmas are in effect only for the `<script>` block they appear in, i.e. they are reset (stack cleared) for each new `<script>` block or module.17

**EXAMPLE**

In this example, nestable comments are turned on. Thus the “print bio of author” comment, which is inside a larger comment, nests properly, and the entire inner `<SQL>` statement is commented out:

```
<!-- pragma nestcomment on -->
<A NAME=main>
  <SQL "select Title, Author from books
       where Keywords like "query">
    Title: $Title <!-- print title -->
    Author: $Author  <!-- print author -->
  </SQL>
  <!-- this section commented out: -->
  <SQL "select Details from author
       where Author = $Author">
    Bio: $Details <!-- print bio of author -->
  </SQL>
</A>
```

**SEE ALSO**

urlcp

---

17 In version 6 and earlier, this was not true; pragmas might be “remembered” across modules or script blocks.
1.4 Special Variables

Certain variables are reserved in Vortex for special uses. Most of these (like $url) are set automatically by Vortex, and should not be assigned to by a Vortex script.

1.4.1 $cmd

The $cmd variable is used to indicate what function to start execution at. If the variable $cmd is set, that function is the start function instead of the default main. A function extension given in the URL, however, overrides this (see p. 8).

1.4.2 $cmdlnargs

The $cmdlnargs variable is set at startup to the list of command-line arguments given to Vortex. It was added in version 2.1.904540000 19980831. See also <vxinfo scriptargs>, p. 294.

1.4.3 $errnum,$errscript,$errline etc.

These variables are set when the putmsg function is called to redirect error messages (see p. 581):

$errnum,$errscript,$errline,$errmsg,$errfunc,$errvfunc

1.4.4 $indexcount

The $indexcount variable is set once, when a SQL statement is executed (e.g. before the first row is returned). It gives an estimate of how many rows will match the given query: its value is the number of matching rows (if determinable) found in the index(es) used by the query. This is different from $next, which is the exact number of returned rows so far (updated every iteration). See the SQL statement (p. 27) for details. The $indexcount variable was added Oct. 24 1996. See also $sqlresult.returnedmin/$sqlresult.returnedmax, which may be more accurate, and set even when no index used. See also $sqlresult.indexcount.

1.4.5 $rows.min,$rows.max

The $rows.min and $rows.max variables are set at every iteration, and at the end, of a <SQL> loop. They are the minimum and maximum total rows that will be returned by the SQL query. In cases where the exact number of result rows is not yet known (e.g. post-processing is required), the two will differ. If the upper or lower limit is not known (e.g. a completely unindexed query), one or both variables will be negative. If both variables are equal, the total row count is exact.

A one-row look-ahead feature is enabled with $rows.min, such that if $rows.min is greater than $next (either in the loop or after), at least one more row exists in the result set.

See p. 33 for more on these variables. See also $sqlresult.... vars, which are more accurate.
### 1.4.6 $sqlresult.... vars

More information about the `<SQL>` query being executed, set at every iteration and at the end of the loop. $sqlresult.returnedmin and $sqlresult.returnedmax are the minimum and maximum number of total (i.e. ignoring SKIP and MAX) result rows that will be returned. In cases where the exact number of result rows is not yet known (e.g. post-processing is required), the two will differ. If the upper or lower limit is not known (e.g. a completely unindexed query), one or both variables will be negative. If both variables are equal, the total row count is exact.

$sqlresult.matchedmin and $sqlresult.matchedmax are similar, but are the min/max total rows matched – i.e. by the WHERE clause, before any reduction by likeprows, GROUP BY etc.

$sqlresult.indexcount is the same as $indexcount, but is set to -1 (not 0) when unknown, to distinguish from 0 (known but zero results).

The $sqlresult.... variables were added in version 6. They are more accurate than the $rows.... variables, as their information is obtained directly from the Texis SQL engine. See p. 33 for more on these variables.

### 1.4.7 $loop

The $loop variable is incremented on every iteration of a LOOP, SQL, TIMPORT, WHILE, READLN or fetch block, starting with 0. For LOOP, it starts with the number of SKIPped values instead. After the loop block ends, $loop is the number of iterations (plus the SKIPped values for LOOPs).

### 1.4.8 $next

The $next variable is set every iteration and at the end of LOOP, SQL, TIMPORT and fetch loops. Inside the loop, it is equivalent to the iteration count, starting from 1. At the end of the loop, it is the value to use for SKIP in a new LOOP or SQL to continue iterating at the next value. For LOOP statements, $next has the same value as $loop (plus 1 inside), but for SQL $next is equal to $loop plus the initial SKIP value (plus 1 inside).

### 1.4.9 $null

$null contains a list of “null” values for parameter substitution in SQL commands. If a single-valued variable used in a `<SQL>` command matches any value of $null, the parameter is not set. This allows unneeded parts of where clauses to be dropped (see the SQL statement, p. 27).

### 1.4.10 $pathroot

The $pathroot variable is similar to the $PATH_INFO environment variable set by the web server, except that it does not contain state or function/MIME-type information (see section 1.1.6 on URL syntax). $pathroot is the URL path (or file path if run from the command line) to the source of the current Vortex
script. It is set once at script startup. The $pathroot variable was added in version 2.1.867816000 19970702.

1.4.11 $ret

The $ret variable is set to the return value of a user-defined or builtin function when called (see p. 89). All variables beginning with the prefix $ret. should be considered reserved for future use as special variables in Vortex.

1.4.12 $ret.count

The $ret.count variable is set to the corresponding counts for an xtree DUMP or SEARCH, in the looping version. All variables beginning with the prefix $ret. should be considered reserved for future use as special variables in Vortex.

1.4.13 $ret.seq

The $ret.seq variable is set to the corresponding sequence numbers for an xtree DUMP or SEARCH, in the looping version. All variables beginning with the prefix $ret. should be considered reserved for future use as special variables in Vortex.

1.4.14 $ret.off

This is set to the corresponding integer byte offset into the current search buffer of the start of the hit, for rex and split, in the looping version. All variables beginning with the prefix $ret. should be considered reserved for future use as special variables in Vortex.

1.4.15 $ret.err, $ret.owner, $ret.group, $ret.size, etc.

These variables are set by the <stat> function (p. 327), and contain various information about the file(s) requested:

$ret.err, $ret.owner, $ret.group, $ret.size, $ret.isrd, $ret.iswr, $ret.isex, $ret.mode, $ret.atime, $ret.mtime, $ret.ctime, $ret.depth, $ret.symlink, $ret.sympath, $ret.nlinks, $ret.devtype, $ret.dev, $ret.ino, $ret.blks, $ret.blksize, $ret.attrib

1.4.16 $sourcepath

The $sourcepath variable is set at startup to the file (not URL) path to the source of the Vortex script (not module) being executed, including “.vs” extension if applicable. It is similar to the $PATH_TRANSLATED variable set by most Web servers, except state/function information has been
1.4. SPECIAL VARIABLES

stripped. The $sourcepath variable was added in version 2.1.904540000 19980831. Note: this is deprecated in favor of <vxinfo sourcepath>, which is more reliable as it cannot be overwritten as a variable can.

1.4.17 $url

The $url variable has the value of the URL to re-invoke the current script, i.e. the URL path to the Vortex executable, plus the script and any state variable information. It is set every time it is printed, to reflect the current value(s) of EXPORTed URL state variables. If no variables are being EXPORTed, or none of them are set, then $url is just a “plain” URL to invoke the script, without state information.

1.4.18 $urlfunc, $urlext

The $urlfunc variable contains the value of the function name part of the URL which was given to invoke the script. Similarly, $urlext contains the MIME filename extension (including the period). If no function/extension was given, these variables are empty. Added in version 3.0.943000000 19991119.

1.4.19 $urlq

The $urlq variable is a URL-encoded query string containing EXPORT QUERY variables. It begins with a “?” character, unless no EXPORT QUERY variables are set, in which case it is empty. It is set every time it is printed, to reflect the current value(s) of EXPORT QUERY state variables. The $urlq variable was added in version 2.1.898900000 19980626.

1.4.20 $urlroot

The $urlroot variable is similar to $url, except that the resulting URL does not contain state information, nor is state-saving invoked. $urlroot is set once at startup, and can be used to obtain a “plain” URL to the script, e.g. that will discard all EXPORTed variables upon invocation. The $urlroot variable was added Mar. 4 1997.

1.4.21 $userpath

The $userpath variable contains the “/userpath” part of the URL that invoked the script, if present (p. 8). The remainder of the URL, after the delimiter “/+”, becomes $userpath. $userpath is set at script start, and must be absolute (i.e. start with /). The $userpath variable was added in version 2.1.873259200 19970903.

1.4.22 $__FILE__

The $__FILE__ variable contains the file path to the source of the currently running script or module, including extension if applicable. If the module is from the Vortex library (e.g. <uses myModule>, the
value is empty. Added in version 6.

1.4.23  $__LINE__$

The $__LINE__$ variable contains the current source script or module line number. Added in version 6.

1.4.24  $__SCRIPT__$

The $__SCRIPT__$ variable contains the name of the current script or module, for messages. I.e. it is the URL path for scripts, or the module name (in square brackets). It may or may not contain a “.vs” or “.vtx” extension, or directory prefix, depending on how the script was invoked. Added in version 6.

1.4.25  $__FUNCTION__$

The $__FUNCTION__$ variable contains the name of the current Vortex function. Added in version 6.
1.5 Builtin Functions

Vortex has many builtin functions that provide additional functionality. Most of these functions take arguments; each argument is either a single variable or a literal string. Each function returns a value in the special variable $\text{ret}$ (except as noted).
1.5.1 sum – return sum of variable values

SYNOPSIS

<sum $fmt $var [$var ...]>

DESCRIPTION

The sum function sums its $var arguments' values – either arithmetically or via string concatenation. The $fmt argument is a <fmt>-style format string that controls how the values are summed, as well as how each value is processed before summing. If $fmt is a string format code (e.g. "%s"), each $var value is printed to a string with $fmt, and the results string-concatenated for a string (varchar) result. If $fmt is a numeric code (e.g. "%d" or "%f"), each $var value is cast to the type indicated by the code (with all integer types promoted to int64), and the results arithmetically summed, for a numeric result of the same type.\footnote{Note that in version 6 (or compatibilityversion 6) and earlier, it was the data ($var) arguments that controlled the overall string-vs-numeric sum behavior: if one or more $var values was non-numeric, string concatenation occurred – regardless of $fmt value – otherwise numeric summation was used. Also, numeric summation used the $fmt string to print the final arithmetic sum – including any significant-digit truncation implied therein – before casting it back to the numeric type for return. This version-6-and-earlier behavior is restored if vxcp compatibilityversion is set to 6.}

DIAGNOSTICS

sum returns the arithmetic sum or string concatenation of its $var arguments, depending on the $fmt code.

EXAMPLE

<$x = 1 2 3>
<sum "%d" $x 4>
$ret
<sum "%10s" "one" "two" "three">
$ret

The output would be (note spacing):

10
one two three
CAVEATS

The \texttt{sum} function was added Sep. 20 1996. In versions prior to Nov 25 1996, the $\texttt{fmt}$ argument was ignored when concatenating strings.

The $\texttt{fmt}$ argument should be appropriate for the values’ type (numeric or string).

SEE ALSO

\texttt{fmt} \texttt{strfmt}
1.5.2 fmt, strfmt – formatted output

SYNOPSIS

<fmt format [arg ...]>

or

<strfmt format [arg ...]>

DESCRIPTION

The fmt function allows flexible control over how variables are printed. It behaves much like the standard C function printf(), with additional formats for URL escapement, query markup, date/time printing, etc. The function strfmt is the string version – its output becomes the value of $ret instead of being printed out.

The format argument controls the output. Each format code in it – a % followed by one or more letters – indicates how to print the next remaining argument. Any escape sequences – a backslash (\) followed by a letter – indicate non-text characters to print. Any other letters in the format string are printed as-is.

DIAGNOSTICS

fmt returns nothing. strfmt returns the formatted output as a string in $ret.

CAVEATS

The fmt function was added Sep. 20 1996. strfmt was added Jan. 30 1997.

Only the first value of multi-value arguments is currently used. Giving too many or too few arguments for the format will result in an error message to that effect.

Escape Sequences

The following escape sequences are recognized in the format string:

- \n Newline (ASCII 10)
- \r Carriage return (ASCII 13)
- \t Tab (ASCII 9)
- \a Bell character (ASCII 7)
• \b Backspace (ASCII 8)
• \e Escape character (ASCII 27)
• \f Form feed (ASCII 12)
• \v Vertical tab (ASCII 11)
• \ Backslash
• \xhh Hexadecimal escape. hh is 1 or more hex digits.
• \ooo Octal escape. ooo is 1 to 3 octal digits.

Standard Formats

A format code is a % (percent sign), followed by zero or more flag characters, an optional width and/or precision size, and the format character itself. The standard format codes, which are the same as in printf(), and how they print their arguments are:

• %d or %i
  Integer number.

• %u
  Unsigned integer number.

• %x or %X
  Hexadecimal (base 16) number; upper-case letters used if upper-case X.

• %o
  Octal (base 8) number.

• %f
  Floating-point decimal number.

• %e or %E
  Exponential floating-point number (e.g. 1.23e+05). Upper-case exponent if upper-case E.

• %g or %G
  Either %f or %e format, whichever is shorter. Upper-case exponent if upper-case G.

• %s
  A text string. The j flag (p. 103) may be given for newline translation.

• %c
  A single character. If the argument is a decimal, hexadecimal or octal integer, it is interpreted as the ASCII code of the character to print. If the ! flag is given, a character is decoded instead: prints the decimal ASCII code for the first character of the argument (added in version 3.01.973800000 20001109).
• %
  A percent-sign; no argument and no flags are given. This is for printing out a literal ‘%’ in the format string, which otherwise would be interpreted as a format code.

A simple example (with its output):

```texis
<fmt "This is %s number %d (in hex: %x)." "test" 42 42>
This is test number 42 (in hex: 2a).
```

**Standard Flags**

After the % sign (and before the format code letter), zero or more of the following flags may appear:

- # (pound sign)
  Specifies that the value should be printed using an “alternate format”, depending on the format code. For format code(s):
  - %o
    A non-zero result will be prepended with 0 (zero) in the output.
  - %x, %X
    A non-zero result will be prepended with 0x or 0X.
  - %e, %E, %f, %g, %G
    The result will always contain a decimal point, even if no digits follow it (normally, a decimal point appears in the results of those conversions only if a digit follows). For %g and %G conversions, trailing zeros are not removed from the result as they would otherwise be.
  - %b A non-zero result will be prepended with 0b.

- 0 (digit zero)
  Specifies zero padding. For all numeric formats, the output is padded on the left with zeros instead of spaces.

- - (negative field width)
  Indicates that the result is to be left adjusted in the output field instead of right. A - overrides a 0 flag if both are present. (For the %L extended code, this flag indicates the argument is a latitude.)

- (a space)
  Indicates that a space should be left before a positive number produced by a signed format (e.g. %d, %i, %e, %E, %f, %g, or %G).

- + (plus sign)
  If given with a numeric code, indicates that a sign always be placed before a number produced by a signed format. A + overrides a space if both are used.
  For the %L extended code, a + flag indicates the argument is a location – latitude and longitude, or geocode.
  If given with a string code, + indicates that if the string value exceeds the given precision, truncate the string by a further 3 bytes, and append an ellipsis (“...”). This can be useful to give an indication of
when a value is being truncated on display. String code support was added in version 6.00.1340835555 20120627.

Examples:

```plaintext
<fmt " %#x  %#x" 42 0>  <fmt "%+d  %+d" 42 -42>
0x2a 0 +42 -42
```

Following any flags, an optional width number may be given. This indicates the minimum field width to print the value in (unless using the m flag; see Metamorph Hit Mark-up, p. 104). If the printed value is narrower, the output will be padded with spaces on the left. Note the horizontal spacing in this example (output is the right column):

```plaintext
Test results: Test results:
<$x = 42 12345 87654321 912> 42
<LOOP $x> 12345
   <$fmt "%6d" $x> 87654321
   912
</LOOP>
```

After the width, a decimal point (.) and precision number may be given. For the integer formats (%d, %i, %o, %u, %x and %X), the precision indicates the minimum number of digits to print; if there are fewer the output value is prepended with zeros. For the %e, %E and %f formats, the precision is the number of digits to appear after the decimal point; the default is 6. For the %g and %G formats, the precision is the maximum number of significant digits (default 6). For the %s (string) format, it is the maximum number of characters to print. Examples:

```plaintext
<fmt "Error number %5.3d:" 5>
Error number 005:

<fmt "The %1.6s is %4.2f." "answering machine" 123.456789>
The answer is 123.46.
```

The field width or precision, or both, may be given as a parameter instead of a digit string by using an * (asterisk) character instead. In this case, the width or precision will be taken from the next (integer) argument. Example (note spacing):

```plaintext
<$width = 10> The value is: 123.46
<$prec = 2> The value is: 123.46
```

An h or l (el) flag may appear immediately before the format code for numeric formats, indicating a short or long value (l has a different meaning for %H, %/ and %:, see p. 101). These flags are for compatibility
with the C function `printf()`, and are not generally needed in Vortex because integers are automatically promoted to the largest type available in version 3.01.984500000 20010312 and later). In these versions, the `w` flag may be given to indicate a “whopping” (largest available, generally 64-bit) value.

**Printing Date/Time Values**

Dates can be printed with `fmt` by using the `%t` format. The `t` code indicates a time is being printed, and the `a` flag indicates that the next argument is a `strftime()`-style format string. Following that is a time argument. Example: `<fmt "%.at" "%B" "now">` where `"%B"` is the `strftime()`-style string (indicating the month should be printed). A capital `T` may be used instead of lower-case `t` to change the timezone to Universal Time (GMT/UTC) instead of local time for output. These `strftime()` codes are available:

- `%a` for the abbreviated weekday name (e.g. Sun, Mon, Tue, etc.)
- `%A` for the full weekday name (e.g. Sunday, Monday, Tuesday, etc.)
- `%b` for the abbreviated month name (e.g. Jan, Feb, Mar, etc.)
- `%B` for the full month name (e.g. January, February, March, etc.)
- `%c` for the preferred date and time representation.
- `%d` for the day of the month as a decimal number (range 01 through 31).
- `%H` for the hour as a decimal number using a 24-hour clock (range 00 through 23).
- `%I` for the hour as a decimal number using a 12-hour clock (range 01 through 12).
- `%j` for the day of the year as a decimal number (range 001 through 366).
- `%m` for the month as a decimal number (range 01 through 12).
- `%M` for the minute as a decimal number (range 00 through 59).
- `%p` for AM or PM, depending on the time.
- `%S` for the second as a decimal number (range 00 through 60; 60 to allow for possible leap second if implemented).
- `%U` for the week number of the current year as a decimal number, starting with the first Sunday as the first day of the first week (range 00 through 53).
- `%W` for the week number of the current year as a decimal number, starting with the first Monday as the first day of the first week (range 00 through 53).
- `%w` for the day of the week as a decimal, Sunday being 0.
- `%x` for the preferred date representation without the time.

`strftime()` is a standard C function for formatting dates.
1.5. BUILTIN FUNCTIONS

- `%X` for the preferred time representation without the date.
- `%y` for the year as a decimal number without a century (range 00 through 99).
- `%Y` for the year as a decimal number including the century.
- `%Z` for the time zone or name or abbreviation.
- `%%` for a literal `%' character.

Since `fmt` arguments are typecast if needed (p. 106), the date argument can be a Texis date or counter type, or a Texis-parseable date string. For example, to print today’s date in the form month/day/year:

```sql
<fmt "%at" "%m/%d/%y" "now">
```

Or to print the title and insertion date of books matching a query, in the style “February 20, 1997” (assuming `id` is a Texis counter field):

```sql
<SQL "select id, Title from books where Desc like $query">
    <fmt "%at" "%B %d, %Y" $id> $Title
</SQL>
```

If the `strftime()` string is composed entirely of `strftime()`-specific codes, none of which have meaning as `fmt` codes, and there is only one argument given, then the initial `%at` may be omitted. For example, to print today’s year, either of these statements may be used:

```sql
<fmt "%t" "now">
<fmt "%Y" "now">
```

The second statement eliminates the `%at`, because `%Y` is a `strftime()`-only code (not valid in `<fmt>`). **Note:** Due to potential confusion and conflicts between `strftime()` and `<fmt>` codes, use of this shorthand is not recommended. Always use the `a` flag and a `t` or `T` code.

To use a default `strftime()` format, eliminate the `a` flag and its corresponding `strftime()` format argument:

```sql
<fmt "%t" "now">
```

This will print today’s date in a default format.

**CAVEATS**

As dates are printed using the standard C library, not all `strftime()` codes are available or behave identically on all platforms.
CHAPTER 1. TEXIS WEB SCRIPT

Printing Latitude, Longitude or Location/Geocode Values

In version 6.00.1300152000 20110314 and later, the %L code may be used with <fmt> to print a latitude, longitude or location (geocode) value, in a manner similar to how date/time values are printed with %t. Flags indicate what type of value is expected, and/or if a subformat is provided:

- (minus)
  A latitude argument is expected (memory aid: latitude lines are horizontal, so is minus sign). This is the default.

| (pipe)
  A longitude is expected (memory aid: longitude lines are vertical; so is pipe).

+ (plus)
  A location is expected; either a geocode long value, or a latitude and longitude (e.g. comma-separated).

a
  Like %at (date/time format), the next argument (before the latitude/longitude/location) is a subformat indicating how to print the latitude and/or longitude. Without this flag, no subformat argument is expected, and a default subformat is used.

Latitude, longitude and location arguments should be in one of the formats supported by the parselatitude(), parselongitude(), or latlon2geocode() (with single arg) SQL functions, as appropriate. If the a flag is given, the subformat string may contain the following codes:

- %D for degrees
- %M for minutes
- %S for seconds
- %H for the hemisphere letter (“H”, “S”, “E” or “W”)
- %h for the hemisphere sign (“+” or “−”)
- %o for an ISO-8859-1 degree sign
- %O for a UTF-8 degree sign
- %% for a percent sign

A field width, precision, space, zero and/or minus flags may be given with the %D/%M/%S codes, with the same meaning as for numeric fmt codes. If no flags are given to a code, the width is set to 2 (or 3 for longitude degrees), with space padding for degrees and zero padding for minutes and seconds.

Additionally, a single d, l, f or g numeric-type flag may be given with the %D/%M/%S codes. This flag will print the value with the corresponding fmt numeric code, e.g. truncated to an integer for d or l, floating-point with potential roundoff for f or g. This flag is only valid for the smallest unit
1.5. BUILTIN FUNCTIONS

(degrees/minutes/seconds) printed: larger units will always be printed in integer format. This ensures that a fractional value will not be printed twice erroneously, e.g. 20.5 degrees will not have its ".5" degrees fractional part printed if "30" minutes is also being printed, because the degrees numeric-type will be forced to integer regardless of flags.

The default numeric-type flag is \texttt{g} for the smallest unit. This helps ensure values are printed with the least number of decimal places needed (often none), yet with more (sub-second) accuracy if specified in the original value. Additionally, for the \texttt{g} type, if a degrees/minutes/seconds value is less than \(10^{-(p-2)}\), where \(p\) is the format code’s precision (default 6), it will be truncated to 0. This helps prevent exponential-format printing of values, which is often merely an artifact of floating-point roundoff during unit conversion, and not part of the original user-specified value.

Examples:

\begin{verbatim}
<!-- Print latitude, default format: -->
<fmt "%-L\n" 41.75>

<!-- Print a longitude in Garmin format: -->
<fmt "%a|L\n" "%H%03D%O%02.2fM" -121.123>

<!-- Dump geocodes for humans: -->
<sql row "select Geocode from geoTable">
  <fmt "Loc: %+L\n" $Geocode>
</sql>
\end{verbatim}

Other Format Codes

In addition to the standard \texttt{printf()} formatting codes, other \texttt{<fmt>} codes are available:

- \texttt{%t, %T}
  \texttt{strftime()}-style output of a date or counter field (see above)

- \texttt{%L}
  Output of a latitude, longitude, or location (geocode); see above

- \texttt{%H}
  Prints its string (e.g. \texttt{varchar}) argument, applying HTML escape codes where needed to make the string “safe” for HTML output (", \& < >, DEL and control chars less than 32 except TAB, LF, FF and CR are escaped). With the ! flag, decodes instead (to ISO-8859-1); see also the \texttt{l (el)} flag, p. 101. The \texttt{j} flag (p. 103) may be given for newline translation. When decoding with !, out-of-ISO-8859-1-range characters are output as ?; to decode HTML to UTF-8 instead, use \texttt{%hV}.

- \texttt{%U}
  Prints its string argument, encoding for a URL, e.g. using %-codes. With the ! flag, decodes instead; with the \texttt{p (path)} flag, spaces are encoded as "%20" instead of "+"; with the \texttt{q} flag, slash and at-sign are encoded as well (or only unreserved/safe chars are decoded, if ! too). See Extended Flags, p. 101.

- \texttt{%V (upper-case vee)}
  Prints its string argument, encoding 8-bit ISO-8859-1 chars for UTF-8 (compressed Unicode). With the ! flag (p. 101), decodes instead (to ISO-8859-1). Illegal, truncated, or out-of-range sequences are
translated as question-marks (?); this can be modified with the \h flag (p. 101). The \j flag (p. 103) may be given for newline translation. Added in version 3.01.970000000 20000926.

* \%v (lower-case vee)
Prints its UTF-8 string argument, encoding to UTF-16. With the ! flag (p. 101), decodes to UTF-8 instead. Illegal, truncated, or out-of-range sequences are translated as question-marks (?); this can be modified with the \h flag (p. 101). The < (less-than) flag forces UTF-16LE (little-endian) output (encode) or treats input as little-endian (decode). The > flag forces UTF-16BE (big-endian) output (encode) or treats input as big-endian (decode). The default endian-ness is big-endian; for decode, a leading byte-order-mark character (hex 0xFEFF) will determine endian-ness if present. The _ (underscore) flag skips printing a leading byte-order-mark when encoding; when decoding the _ flag saves (does not delete) a leading byte-order-mark in the input. The \j flag (p. 103) may be given for newline translation. Added in version 4.03.1049741744 20030407.

* \%B
Prints its string argument, encoding to base64. If a non-zero field width is given, a newline is output after every “width” bytes output (absolute value, rounded up to 4) and at the end of the base64 output. Thus “%64B” would format with no more than 64 bytes per line. This is useful for encoding into a MIME mail message with line length restraints. A ! flag indicates that the string is to be decoded instead of encoded. The \j flag (p. 103) may be given to set the newline style, though it only applies to soft (output) newlines; input CR/LF bytes are never modified since base64 is a binary encoding. Added in version 3.01.984400000 20010312.

* \%Q
Prints its string argument, encoding to quoted-printable (per RFC 2045). If a non-zero field width is given, a newline is output after every “width” bytes output (absolute value, rounded up where needed). A negative field width or – flag indicates “binary” encoding: input CR and LF bytes are also hex-encoded; normally they are output as-is (or subject to the \j flag, p. 103) and therefore subject to possible newline translation by a mail transfer agent etc. A ! flag indicates that decoding instead of encoding is to be done (and the field width and negative flag are ignored). The \j flag (p. 103) may be given for newline translation.

If an underscore (_ flag is given, “Q” encoding (per RFC 2047) is used instead of quoted-printable: it is similar, except that U+0020 (space) is output as underscore (_), no whitespace is ever output (e.g. tab/CR/LF are hex-encoded, and the field width is ignored), and certain other special characters are hex-encoded that normally would not be (e.g. dollar sign, percent, ampersand etc.). With the underscore flag, the resulting output is safe for all RFC 2047 “Q” encoding contexts.

Added in version 4.03.1051320912 20030425.

* \%W
Prints its UTF-8 string argument, encoding linear-whitespace-separated tokens to RFC 2047 encoded-word format (i.e. “=?...?=” mail header tokens) as needed. Tokens that do not require encoding are left as-is. A ! flag indicates that decoding instead of encoding should be done. A \q flag for \%W indicates that only the “Q” encoding should be used for encoded words; normally either Q or base64 – whichever is shorter – is used. The \hh, \hhh, \j, “ and | flags are respected. In version 7.02.1421703000 20150119 and later, the \h flag is supported for \%!W. If a non-zero field width is given, it is used as the desired maximum byte length of encoded words: if an encoded word would be longer than this, it is split atomically into multiple words, separated by newline-space. Added in version 6.00.1283370000 20100901.
1.5. BUILTIN FUNCTIONS

- \%z
  Prints its argument, compressed in the gzip format. The ! flag will gunzip the argument instead. A precision value will limit the output to that many bytes, as in \%s; this can be used to “peek” at the start of compressed data without decoding all of it (and consuming memory to do so). Added in version 7.05.1457041000 20160303.

- \%b
  Binary output of an integer.

- \%F
  Prints a float as a fraction: whole number plus fraction.

- \%r
  Lowercase Roman numeral output of an integer.

- \%R
  Uppercase Roman numeral output of an integer.

- \%/  
  Print platform-specific directory separator, e.g. “/” for Unix and “\ ” for Windows. No argument. Added in version 5.01.1131507000 20051108. With an \l (el) flag, the code instead prints a REX character class (bracketed expression) to match all valid directory separators, e.g. “[/]” for Unix, and “[\\/]” for Windows; this behavior was added in version 7.00.1352409000 20121108, which also added the ! flag to negate the expression. The REX repetition operator is omitted for user flexibility in adding one.\(^\text{20}\)

- \%:
  Print platform-specific search path separator, e.g. “;” for Unix and “;” for Windows. No argument. Added in version 5.01.1131507000 20051108. With an \l (el) flag, the code instead prints a REX character class (bracketed expression) to match all valid search path separators, e.g. “[;]” for Unix, and “[;]” for Windows; this behavior was added in version 7.00.1352409000 20121108, which also added the ! flag to negate the expression. The REX repetition operator is omitted for user flexibility in adding one. (See also %/ footnote.)

All the standard flags, as well as the extended flags (below), can be given to these codes, where applicable. Examples:

```fmt "Year %R %H %R" 1977 "<" 1997
Year MCMLXXVII &lt; MCMXCII
<fmt "%F" 5.75>
5 3/4
```

Extended Flags

The following flags are available for fmt codes, in addition to the standard printf() flags described above:

\(^{20}\) The \l flag was actually added to the / and : codes in version 5.01.1270586000 20100406, but a bracketed class was not always printed, and a repetition operator was printed.
• a
  Next argument is \texttt{strftime()} format string; used for \%t/\%T time code (p. 96).

• k
  For numeric formats, print a comma (,) every 3 places to the left of the decimal (e.g. every multiple of a thousand).

• K (upper case ‘K’)
  Same as k, but print the next argument instead of a comma.

• \& (ampersand)
  Use the HTML entity \&nbsp; instead of space when padding fields. This is of some use when printing in an HTML environment where spaces are normally compressed when displayed, and thus space padding would be lost. Added in version 2.6.931500000 19990709.

• ! (exclamation point)
  When used with \%H, \%U, \%V, \%B, \%C, \%W or \%2, decode appropriately instead of encoding. Added in version 3.01.969000000 20000914. (Note that for \%H, only ampersand-escaped entities are decoded; for parsing and removal of tags see \texttt{fetch}, p. 179.)

• _ (underscore)
  Use decimal ASCII value 160 instead of 32 (space) when padding fields. This is the ISO Latin-1 character for the HTML entity \&nbsp; . Added in version 2.6.931500000 19990709. For the \texttt{”%v”} (UTF-16 encode) format code, a leading BOM (byte-order-mark) will not be output. For the \texttt{”%!v”} (UTF-16 decode) format code, a leading BOM in the input will be preserved instead of stripped in the output. For the \texttt{”%Q”/”%!Q”} (quoted-printable encode/decode) format codes, the “Q” encoding will be used instead of quoted-printable.

• ^ (caret)
  Output only XML-safe characters; unsafe characters are replaced with a question mark. Valid for \%V, \%=V, \%!V, \%v, \%!v, \%W, \%!W and \%s format codes (text is assumed to be ISO-8859-1 for \%s). XML safe characters are all characters except: U+0000 through U+0008 inclusive, U+000B, U+000C, U+000E through U+001F inclusive, U+FFFE and U+FFFF. Added in version 5.01.1220392000 20080902.

• = (equal sign)
  Input encoding is “equal to” (the same) as output encoding, i.e. just validate it and replace illegal encoding sequences with “?”. Unescaping of HTML sequences in the source (h flag) is disabled. Valid for \%V format code. Added in version 5.01.1220402000 20080902.

• | (pipe)
  Interpret illegal encoding sequences in the source as individual ISO-8859-1 bytes, instead of replacing with the “?” character. When used with \%=V for example, this allows UTF-8 to be validated and passed through as-is, yet isolated ISO-8859-1 characters (if any) will still be converted to UTF-8. Valid for \%!V, \%=V, \%v, \%W and \%!W format codes. Added in version 5.01.1220406000 20080902.

• h
  For \%!V (UTF-8 decode) and \%v (UTF-16 encode): if given once, HTML-escapes out-of-range (over 255 for \%!V, over 0x10FFFF for \%v) characters instead of replacing with ? . For \%V (UTF-8 encode)
and `%!v` (UTF-16 decode): if given once, unescapes HTML sequences first; this allows characters that are out-of-range in the input encoding to be represented natively in the output encoding.

For `%V`, `%!V`, `%v`, `%!v`, `%w` and `%!w`, if given twice (e.g. `hh`), also HTML-escapes low (7-bit) values (e.g. control chars, `<`, `>`) in the output. Added in version 3.01.969000000 20000914. (The `h` flag is also used in another context as a sub-flag for Metamorph mark-up, p. 104.) In version 6.00.1335996839 20120502 and later, if given three times (e.g. `hhh`), just HTML-escapes 7-bit values; does not also decode HTML entities in the input.

- `j` (jay)
  For the `%a`, `%H`, `%v`, `%V` and `%Q` format codes (and their `-decode` variants), also do newline translation. Any of the newline byte sequences CR, LF, or CRLF in the input will be replaced with the machine-native newline sequence in the output, instead of being output as-is. This allows text newlines to be portably “cleaned up” for the current system, without having to detect what the system is. If `c` is given immediately after the `j`, CR is used as the output sequence, instead of the machine-native sequence. If `l` (el) is given immediately after the `j`, LF is used as the output sequence. If both `c` and `l` are given (in either order), CRLF is used. The `c` and `l` subflags allow a non-native system’s newline convention to be used, e.g. by a web application that is adapting to browsers of varying operating systems. Note that for the `%B` format code, input CR/LF bytes are never translated (since it is a binary encoding); `j` and its subflags only affect the `output` of “soft” line-wrap newlines that do not correspond to any input character. Added in version 4.03.1056420269 20030623.

- `l` (el)
  For `%H`, only encode low (7-bit) characters; leave characters above 127 as-is. This is useful when HTML-escaping UTF-8 text, to avoid disturbing multi-byte characters. When combined with `!` (decode), escape sequences are decoded to low (7-bit) strings, e.g. “&copy;” is replaced with “©” instead of ASCII character 169. (The `l` flag is also used with numeric format codes to indicate a long integer or double, and with the `j` flag as a subflag.) Added in version 3.01.969000000 20000914. The `l` flag has yet another meaning when used with the `/` or `:` format codes; see discussion of those codes above.

- `m`
  For the `%a`, `%H`, `%V` and `%v` codes, mark up with a Metamorph query. See next section for a discussion of this flag and its subflags `b`, `B`, `h`, `n`, `p`, `P`, `c` and `e`.

- `p`
  Perform paragraph markup (for `%s` and `%H` codes). Paragraph breaks (text matching the REX expression “$=\space+”) are replaced with “<p>” tags in the output. For the `%U` code, do path instead of query-string escapement: space is encoded as `%20` instead of `. `PP` was added in version 6.

- `Q`
  For the `%U` code, do full-encoding: encode “/” (forward slash) and “@” (at-sign) as well. Implies `p` flag as well. Added Dec. 2 1998. For `%!U` (URL decode), only decode unreserved (per RFC 2396

---

21 Unless some other translation is already indicated for CR and/or LF, such as hex escape for “Q.”
section 2.3) characters: alphanumeric, dash, underscore, period, exclamation point, tilde, asterisk, single-quote, left and right parentheses – this was added in version 7.04.1444076000 20151005. For the %W code, only the “Q” encoding will be used (no base64).

Example:

```
<fmt "You owe $%10.2kf to us." 56387.34>
You owe $ 56,387.34 to us.
```

Metamorph Hit Mark-up

The %s, %H, %V and %v <fmt> codes can execute Metamorph queries on the string argument and mark-up the resulting hits. An m flag to these codes indicates that Metamorph hit mark-up should occur; the Metamorph query string is then taken to be the next argument (before the normal string argument to be searched and printed). The m flag and its sub-flags are only valid for the %s and %H codes.

Following the m flag can be any of the following sub-flags. These must immediately follow the m flag, as some letters have other meanings elsewhere:

- I for inline stylesheet (<span style=...>) highlighting with different styles per term
- C for class (<span class=...>) highlighting with different classes per term
- b for HTML bold highlighting of hits
- B for VT100 bold highlighting of hits
- U for VT100 underline highlighting of hits
- R for VT100 reverse-video highlighting of hits
- h for HTML HREF highlighting (default)
- n indicates that hits that overlap tags should not be truncated/moved
- p for paragraph formatting: print “<p/>” at paragraph breaks
- P same as p, but use (next additional argument) REX expression to match paragraph breaks. If given twice (PP), use another additional argument after REX expression as replacement string, instead of “<p/>”. PP was added in version 6.
- c to continue hit count into next query call
- N to mark up NOT terms as well
- e to mark up the exact query (no queryfixupmode/NOT processing)
- q to mark up the query itself, not the text, e.g. as a legend

22Before version 5.01.1138398232 20060127, NOT terms were marked up by default (without the N flag).
For example, to highlight query terms from $query$ in the text contained in $buffer$ in different colors, insert paragraph breaks, and escape the output to be HTML-safe, use:

```
<fmt "%mlpH" $query $buffer>
```

Each hit found by the query has each of its sets’ hits (e.g. each term) highlighted in the output. With I and/or C highlighting, if there are delimiters used in the query, the entire delimited region is also highlighted. The Metamorph query uses the same apicp defaults and parameters as SQL queries. These can be changed with the apicp function (p. 114).

If a width is given for the format code, it indicates the character offset in the string argument to begin the query and printing (0 is the first character). Thus a large text argument can be marked up in several chunks. Note that this differs from the normal behavior of the width, which is to specify the overall width of the field to print in. The precision is the same – it gives the maximum number of characters of the input string to print – only it starts counting from the width.

The h flag sets HREF highlighting (the default). Each hit becomes an HREF that links to the next hit in the output, with the last hit pointing back to the first. In the output, the anchors for the hits are named hit\(N\), where \(N\) is the hit number (starting with 1).

Hits can be bold highlighted in the output with the b flag; this surrounds them with <b> and </b> tags. b and h can be combined; the default if neither is given is HREF highlighting. In version 5.01.1212100000 20080529 and later, the B and U flags may be given, for VT100-terminal bold and underline highlighting; this may be useful for command-line scripts. In version 6.00.1297382538 20110210 and later, the R flag may be given for VT100-terminal reverse-video highlighting.

In version 6 and later, the I or C flags may be given, for inline styles or classes. This allows much more flexibility in defining the markup, as a style or class for each distinct query term may then be defined. The styles and classes used can be controlled with <fmtcp> (p. 107).

In version 5.01.1223065000 20081003 and later, the q flag may be given, to highlight the query itself, instead of the following text buffer (which must still be given but is ignored). This can be used at the top of a highlighted document to give a highlighting “legend” to illustrate what terms are highlighted and how.

The n and e flags are also implicitly enabled when q is given. Note that settings given inline with the query (e.g. "@suffixproc=0") will not be highlighted (in version 6.00.1316840000 20110924 and later), since they do not themselves ever find or match any terms – this helps avoid misleading the user that such “terms” will ever be found in the text. However, since they are still considered separate query sets – because their order in the query is significant, as they only affect following sets – a class/style is “reserved” (i.e. not used) for them in the querycyclenum rotation.

Normally, hits that overlap HTML tags in the search string are truncated or moved to appear outside the tag in the output, so that the highlighting tags do not overlap them and muddle the HTML output. The n tag indicates that this truncation should not be done. (It is also not done for the %H (HTML escapement) format code, since the tags in the string will be escaped already.)

The p and P flags do paragraph formatting as documented previously.

The c flag indicates that the hit count should be continued for the next query. By default, the last hit marked up is linked back to the first hit. Therefore, each %-code query markup is self-contained: if multiple calls are made, the hit count (and resulting HREFs) will start over for each call, which may not be desired. If the c
flag is given, the last hit in the string is linked to the “next” hit \((N + 1)\) instead of the first, and the next query will start numbering hits at \(N + 1\) instead of 1. Thus, all but the last query markup call by a script should use the \(c\) flag.

The \(e\) flag indicates that the query should be used exactly as given. Normally, \texttt{queryfixupmode} (p. 109) and \(N\) flag processing is done to the query, which might cause more terms to be highlighted than are actually found by the query (e.g. highlighting of sets in the query that are not needed to resolve it, if not all sets are required). With \(e\) set, such processing is not done, and some apparent hits may be left unhighlighted. This processing and the \(e\) flag were added in version 2.00.897097720 19980605. See \texttt{queryfixupmode} (p. 109) for details on how the query is modified when \(e\) is not given.

The following example marks up each \$body value from a table that matches the user’s submitted \$query string. Each set (term) is color-coded differently, and the \$body text is HTML-escaped:

```sql
<sql max=10 "select body from data where body like $query">
  <fmt "%mIH" $query $body>
</sql>
```

**Automatic Typecasting**

Unlike C’s \texttt{printf()}, the arguments to \texttt{fmt} are automatically cast to the required type if possible. Thus, a floating-point number can be printed as a hexadecimal integer with “\%X”, or a date field URL-escaped with “\%U”. Any cast that is nonsensical or impossible (like a \texttt{varbyte} field to “\%t” time) will print a “?”.

In version 3.01.984500000 20010313 and later, integer formats are automatically cast to the largest integer type available to avoid truncation, so the \(l\), \(ll\) or \(w\) flags are not generally needed. This behavior can be controlled with the \texttt{promoteints} argument to \texttt{fmtcp} (p. 107).

**CAVEATS**

Many typecasts are possible but result in truncated output; e.g. printing a float as an int will truncate any decimal places off.

**SEE ALSO**

\texttt{fmtcp}, \texttt{mm}
1.5. BUILTIN FUNCTIONS

1.5.3 fmtcp – set fmt control parameters

SYNOPSIS

<fmtcp $setting [args ...]>

DESCRIPTION

fmtcp controls various facets of fmt and variable printing behavior. The possible settings are:

- **query $fmt $query**
  Sets the Metamorph query to use for automatic hit markup (highlighting of user query terms in the document). Takes a fmt-style format for Metamorph markup and the query as arguments, e.g.
  `<fmtcp query "%mIH" $query>`.
  After this call, the `<mm>` and `</mm>` functions (p. 111) may be used to enable the automatic markup of all variables printed within such a `<mm>` block, using the given query and format.
  For marking up just a single variable, it may be easier to call `<fmt>` directly (p. 104).

- **sandblast [noesc] $search $replace**
  Sets the search and replace list to use for automatic search and replace. Takes a list of REX search expressions and a list of corresponding plain-text replacement strings, like `sandr` (p. 155) with the difference that no characters in the replace string are special. After this call, the `<sb>` and `</sb>` functions (p. 113) may be used to enable the automatic search and replace of all variables printed, using these expressions.
  If the optional noesc flag is given, then HTML escapement of variables after replacement is **not done** in HTML mode, as it would normally be done. In non-HTML content types noesc has no effect.
  sandblast was added in version 2.1.899870000 19980708.

- **sandcall [noesc] $search $func**
  Like sandblast, but with callback functions instead of replace strings. When a REX expression in the $search list matches, the corresponding Vortex script function named in $func is called, instead of printing a replacement string. The function can have the following parameters which will be set:
  
  - **hit**
    The string matching the REX expression.
  
  - **expn**
    The index of the expression which matched, starting with 0 for the first expression in $search.
  
  - **tag**
    The name of the first HTML tag recognized in the hit.
  
  - **attrs**
    The attributes of the first HTML tag in the hit.
- **vals**
  The corresponding values of the attributes of the first tag in the hit.

The `tag`, `attrs` and `vals` parameters are useful for on-the-fly parsing of HTML embedded within some source data. If `noesc` is given, then HTML escapement of the non-hit portions of variables is not done in HTML mode, as would be the default.

`sandcall` was added in version 2.6.918680000 19990210.

Note that in version 7.06.1518116000 20180208 and later, callback parameter initialization became stricter; see the discussion for `<pagelinks>` callbacks (p. 135) for details.

- **promoteints on|off**
  Controls whether to automatically promote integer format codes in `fmt` and `strfmt` to the largest type available, i.e. implicitly use the `w` flag. With this on, large integers can be printed without truncation without having to specify the `l`, `ll`, or `w` flags in the format code. On by default. Added in version 3.01.984500000 20010312.

- **defaults**
  Restores default settings; also clears `sandblast/sandcall` settings. Added in version 5.01.1223065064 20081003.

- **querystyle $style|default**
  Sets the inline style for highlighting the entire query, when a delimiter (e.g. `w/sent`) is used. Setting `default` restores the default, which is bold black text on light grey. If an empty string is set, or no delimiter is used in the query, or the delimiter is the whole document, no overall-query style highlighting is done (but see `highlightwithin`doc). Inline styles are used with the `I` highlighting subflag of `<fmt>`. Added in version 6.

- **querysetstyles [$styles ...]|default**
  Sets the inline styles for highlighting individual sets (terms) in a query. Setting `default` restores the default, which is white or black text on 10 different background colors. If the number of styles set is less than `querysetcyclenum`, the last style will be re-used for later terms. If an empty string is set, no per-set style highlighting is done. Inline styles are used with the `I` highlighting subflag of `<fmt>`. Added in version 6.

- **queryclass $class|default**
  Sets the class name for highlighting the entire query, when a delimiter (e.g. `w/sent`) is used. Setting `default` restores the default, which is “query”. If an empty string is set, or no delimiter is used in the query, or the delimiter is the whole document, no overall-query class highlighting is done (but see `highlightwithin`doc). Note that it is up to the programmer to supply a stylesheet definition for the `query` class and any other set classes used; see `<fmtinfo>` (p. 110). Inline classes are used with the `C` highlighting subflag of `<fmt>`. Added in version 6.

- **querysetclasses [$classes ...]|default**
  Sets the class names for highlighting individual sets (terms) in a query. Setting `default` restores the default, which is “queryset1”. If the number of classes set is less than `querysetcyclenum`, the last class name is re-used, but the first integer substring within it is incremented; e.g. “queryset1” becomes “queryset2”; thus consecutively-numbered class names do not all have to be listed. If an empty string is set, no per-set class highlighting is done. Classes are used with the `C` highlighting subflag of `<fmt>`. Added in version 6.
1.5. BUILTIN FUNCTIONS

- **queryclassesprefix** 
  $prefix|default

  Sets all highlighting classes (entire-query plus set classes) to the concatenation of $prefix and each class’s default name. Provides a single way to ensure all class names are distinct. If “default” is set, uses the default prefix, which is none (i.e. resets all class names to defaults). Added in version 7.03.

- **querysetcyclenum** 
  $n|default

  Sets maximum number of distinct sets (terms) in the query to highlight before recycling styles or classes. The default (which can be restored by setting default) is 10, which means that a query with more than 10 terms will re-use highlighting colors: terms 11-20 will be highlighted the same as terms 1-10, etc. Setting 0 is infinite, i.e. styles will not be recycled. If the cycle num exceeds the number of defined querysetstyles, the last style is re-used. If the cycle num exceeds the number of defined querysetclasses, the last class is re-used, but the first integer substring within it is incremented (e.g. “queryset1” becomes “queryset2”): this provides an easy way to automatically number classes. Added in version 6.

- **highlightwithindoc** 
  on|off

  Whether to use the overall-query highlighting style/class querystyle/queryclass when the query has no delimiters or the delimiters are the whole document. Default is off, since highlighting the entire document is usually not helpful. Added in version 6.

- **queryfixupmode** 
  withindot|findsets

  How to modify queries during highlighting so that all requested terms are highlighted (even if not all are needed to resolve query). The default is findsets, which will preserve the query and find all occurrences of all non-NOT sets, within delimiters. In version 5 and earlier, the default was withindot, which appended “w/. @0” to the query and turned AND sets (+) into SET sets (=). findsets mode is more accurate, as it preserves the original delimiters and set logic; however withindot is more likely to find all sets regardless of query logic. Added in version 6.

**DIAGNOSTICS**

fmt cp has no effect on $ret.

**CAVEATS**

The fmt cp function was added in version 2.1.896760000 19980602. Some options were added in later versions as noted above.

It is not currently possible to Metamorph-markup and search-and-replace at the same time, e.g. have <mm> and <sb> active simultaneously.

**SEE ALSO**

fmt strfmt, mm, sb, rex split
1.5.4 fmtinfo – get fmt control parameters

SYNOPSIS

<fmtinfo $value>

DESCRIPTION

fmtinfo obtains the current values for various facets of fmt and variable printing behavior. The possible values are:

- stylesheet
  Returns a stylesheet for the classes currently defined for query highlighting; i.e. maps the defined classes to the defined styles. This can be printed in a `<style type="text/css">` block in the `<head>` of the document, or as part of a separate `.css` stylesheet. Added in version 6.

DIAGNOSTICS

fmtinfo returns a value according to its arguments.

CAVEATS

fmtinfo was added in version 5.01.1223065000 20081003
1.5.5 \texttt{mm} – enable/disable automatic hit markup

SYNOPSIS

\begin{verbatim}<mm> ... $variables ... </mm>
\end{verbatim}

DESCRIPTION

\begin{verbatim}<mm> and </mm> enable and disable, respectively, automatic hit markup of printed variables, after a query is set with \texttt{fmtcp} (p. 107). This makes it easier to highlight the user’s query terms across many result variables when printing a given search result, without having to search-and-replace each variable.
\end{verbatim}

DIAGNOSTICS

\begin{verbatim}<mm> and </mm> have no effect on $ret.
\end{verbatim}

In this example, the user’s query is assumed to be in the variable $query, and we are printing one search result from that query, identified by $id. We use \begin{verbatim}<mm>\end{verbatim} to auto-highlight the user’s search terms across several fields (title, subject, description, etc.):

EXAMPLE

\begin{verbatim}<fmtcp query "%mIH" $query>
Your query returned:
<SQL "select Title, Subject, Desc from books where id = $id">
<mm>
   Title: $Title\<BR>
   Subject: $Subject \<BR>
   Description: $Desc
<mm></p>
</SQL>
Other references:
<SQL "select Other, DontMark from sometable where id = $id">
<mm>$Other\</mm> $DontMark
</SQL>
\end{verbatim}

Note that if just a single variable is to be marked up, it may be easier with a single \begin{verbatim}<fmt>\end{verbatim} call (p. 104):

\begin{verbatim}<SQL "select Title from books where id = $id">
   Title: <B><fmt "%mIH" $query $Title></B><BR>
</SQL>
\end{verbatim}
without using <fmtcp>, <mm> and </mm>.

CAVEATS
The mm statement was added in version 2.1.896840000 19980603.

Note that only variables are marked up: literal text is not, so that HTML formatting tags (e.g. <BR>) are not modified.

SEE ALSO
fmtcp, fmt strfmt, sb
1.5. BUILTIN FUNCTIONS

1.5.6 sb – enable/disable automatic search and replace

SYNOPSIS

<sb> ... $variables ... </sb>

DESCRIPTION

<sb> and </sb> enable and disable, respectively, automatic search and replace of printed variables, after a search and replace list is set with the $SANDBLAST or $SANDCALL commands to fmtcp.

DIAGNOSTICS

<sb> and </sb> have no effect on $ret.

EXAMPLE

<$x = "A red rose.">
<fmtcp "sandblast" "red" "blue">
Original: $x
<sb>Replace "red" with "blue": $x</sb>

The output would be:

Original: A red rose.
Replace "red" with "blue": A blue rose.

CAVEATS

The sb function was added in version 2.1.899870000 19980708.

sandr-style replacement strings (e.g. special backslash escapes) are not yet supported.

SEE ALSO

fmtcp, fmt strfmt, mm
1.5.7 apicp – modify Metamorph query control parameters

SYNOPSIS

<apicp $name $value [$value ...]>

DESCRIPTION

The apicp function allows the setting of parameters affecting the behavior of Metamorph and the various like searches (like, likep, liker, like3 etc.). The $name value describes a setting–corresponding to an n_set...() API function–that is set to the following $values. These settings affect both SQL statements and the fmt, strfmt and mm functions.

The $value parameter is interpreted as a boolean value, integer, string, or list of strings, depending on the value of $name. Boolean values are “true”, “yes”, “on” or a non-zero integer for true; the opposite for false. Only string-list values may be passed multiple $value arguments.

Query Protection

The following apicp settings alter the set of query syntax and features that are allowed. Metamorph has a powerful search syntax, but if improperly or inadvertently used can take a long time to resolve poorly constructed queries. In a high-load environment such as a Web search engine this can bog down a server, slowing all users for the sake of one bad search.

Therefore, Vortex is by default highly restrictive of the queries it will allow, denying some specialized features for the sake of quicker resolution of all queries. By altering these settings, script authors can “open up” Texis and Metamorph to allow more powerful searches, at the risk of higher load for special searches.

- alequivs (boolean, off by default)
  If on, allows equivalences in queries. If off, only the actual terms in a query will be searched for; no equivalences. This is regardless of ~ usage or the setting of keepeqvs. Note that the equivalence file will still be used to check for phrases in the query, however. Turning this on allows greater search flexibility, as equivalent words to a term can be searched for, but decreases search speed. Note: In tsql version 5 and earlier the default was on.

- alintersects (boolean, off by default)
  If on, allow use of the @ (intersections) operator in queries. Queries with few or no intersections (e.g. @0) may be slower, as they can generate a copious number of hits. Note: In tsql version 5 and earlier the default was on.

- allinear (boolean, off by default)
  If on, an all-linear query–one without any indexable “anchor” words–is allowed. A query like “/money #million” where all the terms use unindexable pattern matchers (REX, NPM or XPM) is an example. Such a query requires that the entire table be linearly searched, which can be very slow for a table of significant size. Note: In tsql version 5 and earlier the default was on.
If `allinear` is off, all queries must have at least one term that can be resolved with the Metamorph index, and a Metamorph index must exist on the field. Under such circumstances, other unindexable terms in the query can generally be resolved quickly, if the “anchor” term limits the linear search to a tiny fraction of the table. The error message “Query would require linear search” may be generated by linear queries if `allinear` is off.

Note that an otherwise indexable query like “rocket” may become linear if there is no Metamorph index on its field, or if an index for another part of the SQL query is favored instead by Texis. For example, with the SQL query “select Title from Books where Date > ’May 1998’ and Title like ’gardening’” Texis may use a Date index rather than a Title Metamorph index for speed. In such a case it may be necessary to enable linear processing for a complicated query to proceed—since part of the table is being linearly searched.

• `alnot` (boolean, on by default)
  If on, allows “NOT” logic (e.g. the – operator) in a query.

• `alpostproc` (boolean, off by default)
  If on, post-processing of queries is allowed when needed after an index lookup, e.g. to resolve unindexable terms like REX expressions, or like queries with a non-inverted Metamorph index. If off, some queries are faster, but may not be as accurate if they aren’t completely resolved. The error message “Query would require post-processing” may be generated by such queries if `alpostproc` is off. Note: In tsq1 version 5 and earlier the default was on.

• `alwild` (boolean, on by default)
  If on, wildcards are allowed in queries. Wildcards can slow searches because potentially many words must be looked for.

• `alwithin` (boolean, off by default)
  If on, “within” operators (w/) are allowed. These generally require a post-process to resolve, and hence can slow searches. If off, the error message “’delimiters’ not allowed in query” will be generated if the within operator is used in a query. Note: In tsq1 version 5 and earlier the default was on.

• `builtindefaults`

• `defaults`
  Restore all settings to defaults set in the texis.ini [Apicp] section (or builtin defaults for settings not set there).

• `denymode` (string or integer; warning by default)
  What action to take when a disallowed query is attempted:
  
    - `silent` or 0
      Silently remove the offending set or operation.
    
    - `warning` or 1
      Remove the term and warn about it with a `putmsg`-catchable message.
    
    - `error` or 2
      Fail the query.
A message such as “’delimiters’ not allowed in query” may be generated when a disallowed query is attempted and denymode is not silent.

• qmaxsets (integer, 100 by default)
The maximum number of sets (terms) allowed in a query. Added in version 2.6.934800000 19990816. Note: also settable as qmaxterms for back-compatibility with earlier versions.

• qmaxsetwords (integer, 500 by default, unlimited by default in tsql)
The maximum number of search words allowed per set (term), after equivalence and wildcard expansion. Some wildcard searches can potentially match thousands of distinct words in an index, many of which may be garbage or typos but still have to be looked up, slowing a query. If this limit is exceeded, a message such as “Max words per set exceeded at word ’xyz*’ in query ‘xyz* abc’” is generated, and the entire set is considered a noise word and not looked up in the index. A value of 0 means unlimited. Added in version 2.6.934900000 19990817.

In version 3.0.947600000 20000110 and later, the set may only be partially dropped (with the message “Partially dropping term ’xyz*’ in query ’xyz* abc’”) depending on the setting of dropwordmode (which must be set with a SQL set statement). If dropwordmode is 0 (the default), the root word, valid suffixes, and more-common words are still searched, up to the qmaxsetwords limit if possible; the remaining wildcard matches are dropped. If dropwordmode is 1, the entire set is dropped as if a noise word.

Note that qmaxsetwords is the max number of search words, not the number of matching hits after the search. Thus a single but often-occurring word like “html” counts as one word in this context.

Note: In tsql version 5 and earlier the default was unlimited.

• qmaxwords (integer, 1100 by default)
The maximum number of words allowed in the entire query, after equivalence and wildcard expansion. If this limit is exceeded, a message such as “Max words per query exceeded at word ’xyz*’ in query ’xyz* abc’” is generated, and the query cannot be resolved. 0 means unlimited. Added in version 2.6.934900000 19990817. Like qmaxsetwords, this is distinct search words, not hits. dropwordmode also applies here. Note: In tsql version 5 and earlier the default was unlimited.

• qminprelen (integer, 2 by default)
The minimum allowed length of the prefix (non-* part) of a wildcard term. Short prefixes (e.g. “a*”) may match many words and thus slow the search. Note: In tsql version 5 and earlier the default was 1.

• qminwordlen (integer, 2 by default)
The minimum allowed length of a word in a query. Note that this is different from minwordlen, the minimum word length for prefix/suffix processing to occur. Note: In tsql version 5 and earlier the default was 1.

• querysetting (string or integer)
Container for changing all or a group of settings to a certain mode. (Explicit texis.ini [Apicp] settings still apply, as with all non-built in “...defaults” settings). The argument may be one of the following:

  – defaults or 0
    Set Vortex defaults; same as <apicp defaults>.
1.5. BUILTIN FUNCTIONS

- texis5defaults or 1
  Set defaults for Texis (i.e. tsql not Vortex) version 5 and earlier. Some of these defaults are in common with Texis 6 and later:
  * alprefixproc, keepnoise, keepeqvs are off
  * alwild, alnot are on
  * minwordlen 255
  * sdexp/edexp are empty
  * eqprefix set to "builtin"
  * ueqprefix set to "eqvsusr"
  * denymode is "warning"
  * qmaxsets is 100

The rest are different from Texis 6 and later:
  * alpostproc, allinear, alwithin, alintersects, alequivs, alexactphrase are on (instead of off in version 6)
  * qminwordlen, qminprelen are 1 (instead of 2 in version 6)
  * qmaxsetwords is unlimited (instead of 500 in version 6)
  * qmaxwords is unlimited (instead of 1100 in version 6)

- vortexdefaults or 2
  Set Vortex defaults; same as <apicp defaults>.

- protectionoff or 3
  Turn off query protection settings, i.e. set all al... settings on (allowed), exactphrase on, qmin... limits to minimums, qmax... limits to maximum (unlimited), denymode to warning.

Added in Texis version 6.

- texisdefaults
  Restore Texis (as opposed to Vortex) version 5 and earlier default values. Note: This setting is deprecated in Texis version 6 and later (as Texis defaults have changed to match Vortex defaults for consistency), and may be removed in a future release. Set querysettings texis5defaults instead. The texisdefaults setting is still respected, but will cause a warning noting that it is deprecated. If legacy scripts cannot be updated to use querysettings texis5defaults instead, this warning can be silenced with the texis.ini setting [Texis] Texis Defaults Warning = off (p. 579).

Setting texisdefaults turns off query protection, e.g. it will enable linear searches, post-processing, within operators, etc. Note: this will permit some queries to run than can potentially take an inordinate amount of time, even with a Metamorph index. Use with caution.

Query Processing

These <apicp> settings affect how a query is processed, e.g. what documents it will match.

- defsuffixm aka defsufrm (boolean, on by default)
Whether to remove a trailing vowel, or one of a trailing double consonant pair, after normal suffix processing, and if the word is still \texttt{minwordlen} or greater. This only has effect if suffix processing is enabled (\texttt{suffixproc on} and the original word is at least \texttt{minwordlen} long). Added in version 3.0.941600000 19991102.

- \texttt{edexp} (string, empty by default)
  The default end delimiter expression.

- \texttt{eqprefix} (string)
  The name of the equivalence file. Default is \texttt{builtin}, which uses the built-in equivalence list.

- \texttt{exactphrase} (tri-state, off by default, on by default in \texttt{tsql})
  Whether to exactly resolve the noise words in phrases. If on, a phrase such as “state of the art” will only match those exact words; however this may require post-processing to resolve the noise words “of the” (potentially slower). If off, any word is permitted in place of the noise words, and no post-processing is done: faster but potentially less accurate. In version 5.01.1178072161 20070501 and later, may be set to \texttt{ignorewordposition}: same as off, but non-noise words are permitted in any order or position; essentially emulates behavior of a non-inverted Metamorph index with no post-processing, but on a Metamorph inverted index too. Note: In \texttt{tsql} version 5 and earlier the default was on.

- \texttt{inced} (boolean, on by default)
  Whether to include the end delimiters in hits. Ignored for \texttt{w/N} (within \texttt{N} chars or words) delimiters.

- \texttt{incsd} (boolean, off by default)
  Whether to include the start delimiters in hits. Ignored for \texttt{w/N} (within \texttt{N} chars or words) delimiters.

- \texttt{intersects} (integer, -1 by default)
  The default number of intersections (if not given in a query).

- \texttt{keepeqvs} (boolean, off by default)
  Whether to use equivalences for words/phrases found in the equivalence file(s) or not.

- \texttt{keepnoise} (boolean, off by default)
  Whether to preserve noise words in the query during processing.

- \texttt{minwordlen} (integer, 255 by default)
  The minimum word length for prefix/suffix processing to occur. Note that this is different from \texttt{qminwordlen}, which is the minimum word length allowed.

- \texttt{noise} (list)
  The noise word list used during query processing. The default noise list is:
• **olddelim** (boolean, off by default)

Whether to emulate “old” delimiter behavior. If turned on, it is possible for a hit to occur outside dissimilar start and end delimiters, such as in this example text:

```plaintext
start-delim ... end-delim ... hit ... start-delim ... end-delim
```

Here the hit is “within” the outermost start and end delimiters, but it’s not within the *nearest* delimiters. With `olddelim` off (the default), this hit now does not match: it would have to occur within the nearest delimiters, which would have to be in the correct order. (Added in version 3.0.950300000 20000211. Previous versions behave as if `olddelim` were on.)

• **phrasewordproc** (string)

Which words of a phrase to do suffix/wildcard processing on. The possible values are `mono` to treat the phrase as a monolithic word (i.e. only last word processed, but entire phrase counts towards `minwordlen`); `none` for no suffix/wildcard processing on phrases; or `last` to process just the last word (default). Note that a phrase is multi-word, i.e. a single word in double-quotes is not considered.
a phrase, and thus \text{phrasewordproc} does not apply. Added in version 4.03.1082000000 20040414. Mode \text{none} supported in version 5.01.1127760000 20050926.

- \text{prefix}(\text{list})
  The prefix list used for prefix processing (if enabled) during search. The default prefix list is:
  \begin{verbatim}
  ante anti arch auto be bi counter de dis em en ex extra fore hyper in inter mis non post pre pro re semi sub super ultra un
  \end{verbatim}

- \text{prefixproc}(\text{boolean, off by default})
  Whether to do prefix processing.

- \text{rebuild}(\text{boolean, on by default})
  Whether to do word rebuilding.

- \text{reqsdelim, reqedelim}(\text{boolean, on by default})
  Whether to require the start (\text{reqsdelim}) or end (\text{reqedelim}) delimiter to actually be present in a hit. If these are turned off, then the given delimiter need not be found for a hit to match; it’s as if the delimiter were “found” at the start or end of the buffer if not present. (Added in version 3.0.9503000000 20000211. Previous versions behave as if these settings were off.)

- \text{sdexp}(\text{string, empty by default})
  The default start delimiter expression.

- \text{see}(\text{boolean, off by default})
  Whether to look up “see also” references during equivalence lookup.

- \text{stringcomparemode}(\text{string})
  Mode and flags for string compares, e.g. equals, less-than etc. It also controls the default mode for most string functions, e.g. \text{<strfold>}, \text{<xtree>} and \text{<sort>}, and the non-case-style flags/mode for the functions \text{lower}, \text{upper} and \text{initcap}. Its value is the same format as the \text{textsearchmode} setting, but the default is “unicodemulti, respectcase” – i.e. characters must be identical to match, though ISO-8859-1 vs. UTF-8 encoding may be ignored. Added in version 6. The version 5 and earlier behavior was effectively “ctype, respectcase, iso-8859-1”. In version 7 (or \text{compatibilityversion} 7) and later, \text{strlst} comparisons also use \text{stringcomparemode}.
  A regular (B-tree) index will always use the \text{stringcomparemode} value that was set at its creation, not the current value. However, when multiple regular indexes exist on the same fields, at search time the Texis optimizer will attempt to use the index whose (creation-time) \text{stringcomparemode} is closest to the current value. This allows some dynamic flexibility in supporting queries with different \text{stringcomparemode} values (e.g. case-sensitive vs. insensitive). \textbf{Caveat: A Texis version 5 or earlier should not access or modify a B-tree index created by a version 6 or later Texis, unless \text{stringcomparemode} was set to “ctype, respectcase, iso-8859-1” at creation, or index corruption may result, especially if there are hi-bit/Unicode/UTF-8 characters.}

- \text{suffix}(\text{list})
  The suffix list used for suffix processing (if enabled) during search. The default suffix list is:
  \begin{verbatim}
  ’ (single quote) able age aged ager ages al ally ance anced ancer ances ant ary at ate ated ater atery ates atic ed en ence enced encer
  \end{verbatim}
• suffixeq(list)
The suffix list used for suffix processing during equivalence lookup. The default suffixeq list is:
' (single quote) ies s

• suffixproc(boolean, on by default)
Whether to do suffix processing.

• textsearchmode(string)
Mode and flags for text searches. This controls case-sensitivity and other character-folding aspects of Metamorph text searches.\(^{23}\) The value consists of a comma-separated list of values: a case-folding style, zero or more optional flags, and a case-folding mode.

The textsearchmode setting may be altered – instead of cleared and set – by using “+” or “-” in front of the given values to denote adding or removing just those values, rather than clearing the whole setting first. This makes it easier to alter just the desired parts, without having to specify the remainder of the setting. E.g. “+respectcase, ignorewidth, -expandligatures” sets the case style to case-sensitive, turns on ignorewidth and turns off ligature expansion, without changing other flags such as ignorediacritics. (Note that negation (“-”) can only be used with values that are “on/off”, i.e. the flags; case style and case mode cannot be negated.) “+” and “-” remain in effect for following values, until another “+”, “-” or “=” (clear the setting first) is given.

The case-folding style determines what case to fold to; it is exactly one of:

- respectcase aka preservecase aka casesensitive
  Do not change case at all, for case-sensitive searches.

- ignorecase aka igncase aka caseinsensitive
  Fold case for caseless (case-insensitive) matching; this is the default style for textsearchmode. This typically (but not always) means characters are folded to their lowercase equivalents.

- uppercase
  Fold to uppercase. Note: This style is for functions that actually return a string, e.g. \(<\text{strfold}>\); it should not be used in comparison situations such as indexes and searches as its comparison behavior is undefined. See the stringcomparemode setting, p. 120.

- lowercase
  Fold to lower-case. Note: This style is for functions that actually return a string, e.g. \(<\text{strfold}>\); it should not be used in comparison situations such as indexes and searches as its comparison behavior is undefined. See the stringcomparemode setting, p. 120.

- titlecase
  Fold to title-case. Titlecase means the first character of a word is uppercased, while the rest of the word is lowercased. Note: This style is for functions that actually return a string, e.g.

\(^{23}\)Currently only index and SPM searches are affected; PPM searches (equivs, parenthetical lists) do not yet support textsearchmode.
<strfold>; it should not be used in comparison situations such as indexes and searches as its comparison behavior is undefined. See the stringcomparemode setting, p. 120.

Any combination of zero or more of the following flags may be given in addition to a case style:

- iso-8859-1 aka iso88591
  Interpret text as ISO-8859-1 encoded. This should only be used if all text is known to be in this character set. Only codepoints U+0001 through U+00FF can be supported. Any UTF-8 text will be misinterpreted.
  If this flag is disabled (the default), text is interpreted as UTF-8, and invalid bytes (if any) are interpreted as ISO-8859-1. This supports all UTF-8 characters, as well as most typical ISO-8859-1 data, if any happens to be accidentally mixed in.\(^\text{24}\)
  Typically, this flag is left disabled, and text is stored in UTF-8, since it supports a broader range of characters. Any other character set besides UTF-8 or ISO-8859-1 is not supported, and should be mapped to UTF-8.

- utf-8 aka utf8
  Alias for negating iso-8859-1, i.e., specifying this disables the iso-8859-1 flag.

- expanddiacritics aka exdiacritics
  Expand certain phonological diacritics:\(^\text{25}\) umlauts over “a”, “o”, “u” expand to the vowel plus “e” (for German, e.g. “f"ur” matches “fuer”); circumflexes over “e” and “o” expand to the vowel plus “s” (for French, e.g. “h"otel” matches “hotel”). The expanded “e” or “s” is optional-match – e.g. “f"ur” also matches “fur” – but only against a non-optional char; i.e. “h"otel” does not match “h"otel” (the “e” and “s” collide), and “fur” does not match “fuer” (both optional “e”\ ’s must match each other). Also, neither the vowel nor the “e”/“s” will match an ignorediacritics-stripped character; this prevents “f"ur” from matching “fu"er”.

- ignorediacritics aka igndiacritics
  Ignore diacritic marks – Unicode non-starter or modifier symbols resulting from NFD decomposition – e.g. diaeresis, umlaut, circumflex, grave, acute, tilde etc.

- expandligatures aka expligatures
  Expand ligatures, e.g. “œ” (U+0153) will match “oe”. Note that even with this flag off, certain ligatures may still be expanded if necessary for case-folding under ignorecase with case mode unicodemulti; see below.

- ignorewidth aka ignwidth
  Ignore half- and full-width differences, e.g. for katakana and ASCII.

Due to interactions between flags, they are applied in the order specified above, followed by case folding according to the case style (upper/lower etc.). E.g. expanddiacritics is applied before ignorediacritics, because otherwise the latter would strip the characters that the former expands.

A case-folding mode may also be given in addition to the above; this determines how the case-folding style (e.g. upper/lower/title) is actually applied. It is one of the following:

\(^{24}\)There is a small chance of misinterpreting ISO-8859-1 data, if adjacent characters coincidentally form a valid UTF-8 sequence. Hence it is preferable that all data be proper UTF-8 when iso-8859-1 is off.

\(^{25}\)The expanddiacritics flag is not currently supported for indexes.
- **unicodemulti**
  Use the built-in Unicode 5.1.0 1-to-N-character folding tables. All locale-independent Unicode characters with the appropriate case equivalent are folded. A single character may fold to up to 3 characters, if needed; e.g. the German *es-zett* character (U+00DF) will match “ss” and vice-versa under `ignorecase`. Note that additional ligature expansions may happen if `expandligatures` is set.

- **unicodemono**
  Use the built-in Unicode 5.1.0 1-to-1-character folding tables. All locale-independent Unicode characters with the appropriate case equivalent are folded. Note that even though this mode is 1-to-1-character, it is not necessarily 1-to-1-byte, i.e. a UTF-8 string may still change its byte length when folded, even though the Unicode character count will remain the same.

- **ctype**
  Use the C `ctype.h` functions. Case folding will be OS- and locale-dependent; a locale should be set with the SQL `locale` property. Only codepoints U+0001 through U+00FF can be folded; e.g. most Western European characters are folded, but Cyrillic, Greek etc. are not. Note that while this mode is 1-to-1-character, it is not necessarily 1-to-1-byte, unless the `iso-8859-1` flag is also in effect. This mode was part of the default in version 5 and earlier.

The default case-folding mode is `unicodemulti`; see below for the version 5 and earlier default, and important caveats.

In addition to the above styles, flags and modes, several aliases may be used, and mixed with flags. The aliases have the form:

```
[stringcomparemode|textsearchmode][default|builtin]
```

where `stringcomparemode` or `textsearchmode` refers to that setting’s value (if not given: the setting being modified). `default` refers to the default value (modifiable with `texis.ini`); `builtin` refers to the built-in factory default; no suffix refers to the current setting value. E.g. “`stringcomparemodedefault,+ignorecase`” would obtain the default `stringcomparemode` setting (from `texis.ini` if available), but set the case style to `ignorecase`.

A Metamorph index always uses the `textsearchmode` value that was set at its initial creation, not the current value. However, when multiple Metamorph indexes exist on the same fields, at search time the Texis optimizer will attempt to use the index whose (creation-time) `textsearchmode` is closest to the current value.

The `textsearchmode` setting was added in Texis version 6; its default is “`unicodemulti, ignorecase, ignorewidth, ignorediacritics, expandligatures`” (note that UTF-8 text is expected, since `iso-8859-1` is not specified in the default). In version 5 and earlier the default was effectively “`ctype, ignorecase, iso-8859-1`”. Caveat: A Texis version 5 or earlier should not access or modify a Metamorph index created by a version 6 or later Texis, unless `textsearchmode` was set to “`ctype, ignorecase, iso-8859-1`” at creation, or index corruption may result, especially if there are hi-bit/Unicode/UTF-8 characters.

- **ueqprefix (string)**
  The name of the user equivalence file. Default is empty.
• withinmode (string)
  A space- or comma-separated unit and optional type for the “within-N” operator (e.g. w/5). The unit
  is one of:
  
  – char for within-N characters
  – word for within-N words

The optional type determines what distance the operator measures. It is one of the following:

  – radius (the default if no type specified when set) indicates all sets must be within a radius N
    of an “anchor” set, i.e. there is a set in the match such that all other sets are within N units right
    of its right edge or N units left of its left edge.
  – span indicates all sets must be within an N-unit span

Added in version 4.03.1081200000 20040405. The optional type was added in version
5.01.1258712000 20091120; previously the only type was implicitly radius. The default setting for
version 5 and earlier is char (i.e. char radius); in version 6 and later the default is word
span.

• withinproc (boolean, on by default)
  Whether to process the w/ operator in queries.

DIAGNOSTICS

apicp returns nothing.

EXAMPLE

<apicp "alpostproc" "on">

CAVEATS

The apicp function was added Sep. 13 1996. Various settings were added since then and are unknown to
previous versions.

Any apicp calls should take place after USER/PASS statements, but before SQL and fmt calls.

The ability to pass multiple $value arguments for string-list settings was added in version 3.0.996300000
20010728.

SEE ALSO

apiinfo, USER PASS, Metamorph hit markup (p. 104)
The Metamorph Linguistics chapter in the Texis manual
1.5.8 apiinfo – get current Metamorph query control parameters

SYNOPSIS

<apiinfo $setting>

DESCRIPTION

The apiinfo function returns the current value(s) for the apicp setting named $setting, which can be any of the valid single-setting apicp settings (e.g. not “defaults” which sets more than one setting). In addition, the value “settings” may be given, which will return a list of the names of all settings.

CAVEATS

The apiinfo function was added in version 6.

SEE ALSO

apicp
1.5.9 sqlcp – modify low-level SQL control parameters

SYNOPSIS

<sqlcp $name $value [...]>

DESCRIPTION

The sqlcp function allows the setting of various low-level and debug parameters affecting the behavior of SQL statements. \( \text{Note: For Metamorph-level query processing (thesaurus, suffixes, etc.) see the apicp function on p. 114.} \) The $name parameter specifies a setting to change, and $value its new value. The settings may also affect SQL expressions in <IF>, <WHILE>, assignment, etc. statements as well. Some settings have additional optional args after $value.

The $value parameter is interpreted as a boolean value, integer, string, or list of strings, depending on the value of $name. Boolean values are "true", "yes", "on" or a non-zero integer for true; the opposite for false. The possible values for $name and what argument(s) they expect are:

- **cache cleanupinterval $n**
  Sets the cleanup interval in seconds for Vortex’s Texis SQL handle cache. Returns 1 on success, 0 on failure, -1 on syntax error. Default interval is 10 seconds. Added in Vortex version 6.00.1306189000 20110523. During cleanup, old/stale handles may be closed if detected.

- **cache close [db|exceptdb $dbList]**
  Closes the Texis SQL handle cache in Vortex. This can be used to ensure the process is not still using a database that may be about to be deleted. Returns 1 on success, 0 on failure. The optional db $dbList arguments were added in version 5.01.1111164819 20050318; if specified, only handles using database(s) given in $dbList will be closed. The optional exceptdb $dbList arguments were added in version 6.00.1293076410 20101222; if specified, all handles except those using database(s) given in $dbList will be closed.

- **cache stats**
  Added in version 3.0.958600000 20000517. Prints SQL handle cache statistics.

- **cache resetstats or cache statsreset**
  Added in version 3.0.958600000 20000517. Clears SQL handle cache statistics. (The format of these statistics is subject to change without notice.)

- **createlocksmethods $methods**
  Sets the methods to try (sequentially) to create the locks structure (global file mapping, shared memory segment, or file, depending on platform) when accessing a database. The $methods list is an order CSV list of one or more of the tokens \texttt{direct} or \texttt{monitor}. \texttt{direct} tries to create the locks directly; \texttt{monitor} requests the Texis Monitor create them. See the [Texis] Createlocks Methods setting in texis.ini for more details; this sqlcp setting supercedes that config setting. Added in version 7.00.1372118000 20130624.
1.5. BUILTIN FUNCTIONS

- `expressioncache close|{maxnum $N}`
  If `close` given, closes the compiled SQL expression cache. If `maxnum $N` given, sets the maximum number of open expressions in the cache to $N$; the default is 20. Note that the overhead for a compiled SQL expression is much smaller than for the equivalent cached SQL handle; in particular, no file, semaphore or shared-mem resources are needed. See the `compilesqlexpressions` pragma (p. 82 for more on compiled SQL expressions. Returns 1 on success, 0 on failure, -1 on syntax error. Added in version 6.01.

- `singleuser` (boolean, off by default)
  If true, single-user mode is set in Textra. This means that one of two conditions must be met at all times:
  - This must be the only process accessing the database. Any action, including selects, inserts, deletes, and updates, is permitted in this case.
  - All processes accessing the database--whether single-user or not--must be “read-only”: only select statements are permitted.

  **Note**: If both of these conditions are violated when single-user mode is in effect, severe database corruption may result. Do not set single-user mode unless you know what you are doing!

  Not only do SQL statements access the database, but so do variables EXPORTed to the state table, the `adminsql` function, other executables like `tsql` etc. All access to a database must be taken into account.

  By guaranteeing no simultaneous writes will ever occur to the database, the normal locking mechanisms in Textra can be bypassed, speeding up read/search access to the database.

  Returns previous setting of `singleuser` (1 or 0). (Note: Version 3.0.947100000 20000105 and earlier always returned 1.) The SQL cache is also closed (reset) by this call.

- `arrayconvert [$func ...] [params|results...] on|off [$type...]`
  Controls whether to convert Vortex arrays (multi-value variables) to multi-value fields and/or vice versa, when passing in and out of Textra SQL expressions. For example, a multi-value `varchar` Vortex variable might be converted to a single `strlst` for Textra. Converting arrays to multi-value fields and back allows lists to be manipulated easier, in their entirety, as `<loop>`ing over the array or hand-computing a merged value can often be avoided. This is especially handy with SQL functions that deal with lists of strings, e.g. the XML API.

  `$func` describes under what functions to do array conversion. It is zero or more of the values `sql`, `timport`, `assign`, `expr`; or `all` (the default) to indicate all functions. `assign` refers to variable assignment via SQL, i.e. in parentheses. `expr` refers to SQL expressions, e.g. in complex `<if>` statements.

  `params` or `results` indicates in which direction to do array conversion: `params` will convert Vortex `$`-variable parameters from arrays to multi-value fields, whereas `results` will convert multi-value result fields to Vortex arrays. The default is both ways.

  `on` or `off` indicates whether to turn the indicated conversion(s) on or off.

  `$type` is a list of zero or more source SQL types to do the conversion for, or `alltypes` to indicate all types. The default is every type, except `char`, `indirect` and `byte` for results (which would
otherwise be split into single characters – usually undesired). For params, arrays of char, indirect and byte will be converted to strlst. Numeric, date, counter and recid params will be converted to the multi-value variable (var...) version of the same type. Arrays of strlst values will be merged into one strlst. Internal types will become a list. All other types cannot be converted and will pass as-is with an error message. For results, char, indirect and byte types will be split into one-char-per-value Vortex arrays (this is not normally enabled). strlst results will be split into Vortex varchar arrays. Numeric, date, counter and recid results will be split into non-var Vortex arrays. Internal types will be split into arrays. All other types cannot be converted.

Alternatively, just default may be set to restore the default values; these are alterable with texis.ini (p. 579). Or just builtin may be set to restore the factory builtin defaults.

Note: params conversion – from Vortex $-variable array to multi-value type – will only occur when the variable has more than one value in the current context. Thus, a single-value variable, or a multi-value variable in a loop context, will not be converted. In Texis version 6 and earlier, it may be useful to set the SQL setting varchartostrlstsep to create (which is already the default in version 7 and later) when inserting Vortex arrays into a strlst table column, as that setting will help convert single-value Vortex string arrays that arrayconvert will otherwise leave alone.

Note: results conversion – from multi-value type to Vortex array – can cause parallel Vortex variables that are assigned in a looping function to become out of sync. E.g. <sql> results might have more than one value added to a variable per row, due to array conversion.

Returns 1 on success, 0 on error. Added in version 6. The default is on for all types, except char, indirect and byte for results. Previous versions did not do array conversion, with the exception of multi-value varchar variable parameters to <sql>, which were converted into a parenthetical comma-separated list (i.e. for Metamorph). See also the metamorphstrlstmode SQL setting for how Metamorph deals with strlst queries, and the varchartostrlstsep SQL setting.

• nulloutputstring (string, “NULL” by default)
  Same as the SQL nulloutputstring property: sets the string to output for SQL NULL values. Note that this is different from the string for zero-integer date values, which is always “NULL”.
  Added in version 7.02.1405382000 20140714. Returns 1 on success, 0 on error.

• tracesql (boolean/integer, 0 by default)
  Debug setting; enables tracing of SQL statements. If greater than zero, SQL statements are printed as informational, <putmsg>-capturable messages whenever executed, including their parameters.
  This setting can be used to trace complex, constructed-on-the-fly SQL statements when debugging scripts, as well as other aspects of SQL engine use. The command-line option -tracesql overrides this (and can be used when it’s not appropriate to edit the script). Returns the previous setting. Added in version 3.0.947100000 20000105. See also the <TRACESQL> directive (p. 76) for details on the various possible values, and the -tracesql command line option (p. 567).

• tracemetamorph (boolean/integer, 0 by default)
  For debugging: trace Metamorph searches. This is an integer value whose bits control various Metamorph tracing messages. The bit flag values are subject to change without notice. Added in version 7.02.1406336000 20140725. Current values:
  - 0x0001 Set/phrase/pattern.matcher object open/close calls
  - 0x0002 findsel() calls
1.5. BUILTIN FUNCTIONS

- 0x0004 inset() rejection
- 0x0008 remorph() checks
- 0x0010 Phrase checks
- 0x1000 getppm() calls
- 0x2000 PPM internal calls (pre-phrase)

Returns previous value.

- tracerowfields (string, empty by default)
  For debugging: trace Texit row reads. This is a CSV list of field(s) to print whenever table rows are read, in the form “table.field[, table.field ...]”. Table and/or field may be “*” for all tables and/or all fields. Added in version 7.02.1406752000 20140730.

- traceidx or traceindex (boolean/integer, 0 by default)
  Debug setting; enables tracing of Metamorph index searches. Unsupported/internal, subject to change without notice. Returns the previous setting. Added in version 3.0.947100000 20000105. See also the –traceidx command-line option.

- tracekdbf (integer)
  Debug setting; traces KDBF calls. Value is a set of bit flags, which may change in a future release; currently defined values are:

  - 0x00000001: After open() / close()
  - 0x00000002: Not used currently
  - 0x00000004: After read()
  - 0x00000008: After write()
  - 0x00000010: After ioctl() (seek-significant)
  - 0x00000020: After ioctl() (other)
  - 0x00000040: Data after read()
  - 0x00000080: Data after write()/ioctl()

At least one of the following control flags must be given to enable the above “after” flags:

- 0x00000100: After user calls
- 0x00000200: After internal calls

These flags issue messages before their action:

- 0x00010000: Before open() / close()
- 0x00020000: Not used currently
- 0x00040000: Before read()
- 0x00080000: Before write()
- 0x00100000: Before ioctl() (seek-significant)
- 0x00200000: Before ioctl() (other)
At least one of the following control flags must be given to enable the above “before” flags:

- 0x10000000: Before user calls
- 0x20000000: Before internal calls

- `tracekdbffile(string)`
  Debug setting; may change in a future release. Controls which file(s) `tracekdbf` applies to; the default if empty/unset is all KDBF files. Value is an optional database directory with optional file, e.g. `database/file.tbl`: if no directory given, the given file is traced in any/all databases; if no file given, all files in just the given directory are traced. The file may also be SYS or USR to indicate all system or all user KDBF files (this may also be given with a directory prefix).

- `kdbfiostats(integer or file, 0 by default)`
  Debug setting; enables tracing of KDBF I/O. Unsupported/internal, subject to change without notice. Returns the previous setting. Added in version 3.01.967500000 20000828. The $value can be “summary”, in which case a summary of KDBF handle opens is printed. If a KDBF file name (without directory prefix) is given, I/O for that particular file is summarized. If 1 is given, I/O for all non-SYS KDBF files is summarized; if 2, for all KDBF files. If the value is bitwise-ORed with 4, then specific KDBF opens and closes are also printed. Hex/octet values may be given in version 4.03.108150000 20040409 and later.

- `verbose [sql|assign|expr|other|all ...] N|default`
  Debug setting; increases verbosity in SQL engine to level N, or default level if default given. Verbosity applies to SQL usage specified by zero or more tokens given before level:

  - sql <SQL> statements
  - assign Variable assignments
  - expr <if>/<while>/etc. expressions
  - other Other usage
  - all All of the above

If no usage tokens are given, all is assumed. Added in version 6.00.1330636000 20120301.

- `autocreatedb` or `autocreate` (boolean, on by default)
  Whether to attempt to automatically create a database when needed for SQL expressions in variable assignment, `<IF>` and `<WHILE>` statements (but not `<SQL>`). If the current database is needed for these statements but cannot be opened, Vortex attempts to create it if `autocreatedb` is on, which may be a problem in some circumstances. With `autocreatedb` off, the database is not automatically created and such statements fail if the database does not exist. Added in version 3.01.963600000 20000714. Returns the previous setting.

- `lookahead` (boolean, on by default)
  Whether to do a one-row look-ahead for SQL `select` statements. Normally the next result row beyond the current one being delivered is fetched at each loop iteration, so that it’s always known
whether there’s at least one more row, even for unindexed queries. However, in some instances this may be undesired, e.g. to save the processing time on one extra row when there’s a small MAX limit, \texttt{$\text{rows.min}$}/\texttt{$\text{sqlresult...min}$} isn’t needed, and/or the query is unindexed. (Non-select statements never do look-ahead, to avoid doing more deletes/updates than the programmer may expect.) Added in version 3.0.958600000 20000517. Returns the previous setting. In version 7.05 and later, the lookahead value at \texttt{<sql>} statement start is used throughout that \texttt{<sql>} loop; in previous versions changing the lookahead value inside the loop could have deleterious side-effects (e.g. early termination of the loop).

- \texttt{rmlocks [force] [verbose] $db}
  Removes any stale locks on database \texttt{$db$}. If the \texttt{force} option is given, all locks are removed, and the lock structure (including shared memory segment if applicable) is removed. This can be used prior to removing a database to clean up any attached resources. \textit{Note:} Removing locks on an active database can cause data corruption. Returns 1 if successful, 0 if not. Added in version 3.01.985400000 20010323.

- \texttt{addtable $file [$tbname [$db [$com [$user [$pass [$bits]]]]]]}
  Adds a raw table \texttt{$file$} to the database, like the command-line program \texttt{addtable} (see the Texis manual). This can be used when manually copying a .tbl file from one database to another, to register it with Texis for SQL access. The default SQL name for the table will be derived from the root name of \texttt{$file$}, unless the \texttt{$tbname$} parameter is given: e.g. if \texttt{$file$} is “mybooks.tbl”, the table name defaults to “mybooks”. The table will be added to the current (\texttt{<DB>}) database, unless the \texttt{$db$} argument is given. A comment for SYSTABLES can be provided with the \texttt{$com$} argument, and the table will be owned by \texttt{$user$} (default \texttt{PUBLIC}). \textbf{Note: The source table must have been created by the same platform type as the destination running \texttt{addtable}: the platform is printed in parentheses by \texttt{texis -version}. Do not manually copy a Texis file while it is being modified. Returns 1 if successful, 0 if not. Added in version 3.0.990500000 20010521.

  The \texttt{$nbits$} argument (version 4.01.1030378283 20020826 and later) indicates the source file bit-size of \texttt{$file$}, if it differs from the current Texis version. The file bit-size is indicated by the 4th dash-separated value in parentheses printed by \texttt{texis -version}. Setting this parameter allows a table produced by the same platform Texis (but a different file bit-size) to be adapted to the destination file bit-size. (All other values in the platform string should otherwise be identical.)

- \texttt{copydbf $src $dest [$skip [$max]]}
  Copies as much valid data as possible from Texis KDBF file \texttt{$src$} and appends to file \texttt{$dest$}, creating it if it does not exist, like the command-line program \texttt{copydbf} (see the Texis manual). This can be used to fix a corrupted table (see also the \texttt{kdbfchk} program in the Texis manual), or to compress a table by removing free space. The \texttt{$dest$} file may be the same as \texttt{$src$}, in which case the file will be overwritten in place; note however that if the copy fails, the source data will also be corrupted. If the \texttt{$skip$} argument is given, that many blocks of initial data are skipped on input before copying begins. If \texttt{$max$} is given, at most that many blocks (after \texttt{$skip$}) are copied. \textbf{Note: The source file must be from a machine of the same platform as its destination:} the platform is printed in parentheses by \texttt{texis -version} and should be identical on both machines. \textbf{Do not copy a Texis file while it is being modified, as no locking is used by \texttt{copydbf}. All indexes involving the destination file, if it is a table, must be dropped and re-created.} \textit{Note:} There is a leading KDBF block in table files before any SQL rows; add 1 to \texttt{$skip$/}$max$ as appropriate to included it if needed. Returns 1 if successful, 0 if not. Added in version 3.0.990600000 20010522.
DIAGNOSTICS

sqlcp returns a setting-specific value.

EXAMPLE

<sqlcp cache close>

CAVEATS

The sqlcp function was added in version 2.1.905400000 19980910.

As stated above, single-user mode is dangerous. A sqlcp call to set single-user mode must take place at the start of a script, before any Texis handles are open.

The addtable and copydbf commands involve low-level manipulation of Texis files, and should be used with care.

No sqlcp calls should take place inside a SQL loop.

Additional control parameters are settable via a "set var=value" SQL statement; see the Texis manual for details.

SEE ALSO

apicp, SQLCACHE
1.5. BUILTIN FUNCTIONS

1.5.10 pagelinks – paginate SQL results

SYNOPSIS

<pagelinks [options]>

DESCRIPTION

The pagelinks function automatically paginates the results of a SQL search, printing the necessary HTML and links for previous/next buttons etc. It eliminates the tedious code needed to compute SKIP values and loop over numbered-page links.

The values of $rows.min and $rows.max (p. 33) are used to compute the necessary page link information. Thus, pagelinks can be called after the SQL loop ends, or at the very first row (though possibly with a different accuracy, if the true row count is not known yet).

When called with no options, pagelinks prints out an “X through Y of Z documents” summary, previous- and next-page links, and numbered page-by-page links in a default format. In order to compute these links, the associated <SQL> statement’s SKIP and MAX must be known, so that skips and page sizes can be set properly. Thus pagelinks automatically uses the last <SQL> statement’s SKIP var and MAX value. To ensure that the links work properly, it is the programmer’s responsibility to EXPORT the necessary variables for the original search, such as the SKIP variable and any <SQL> query parameters.

Since pagination design varies widely by application, pagelinks has a number of options to fully control output formatting, as well as defaults for use in simple “quick and dirty” pagination:

- **SKIPVAR**
  The SKIP variable used by the SQL query. This is not simply the SKIP value, but the variable name itself: not only is the current SKIP value obtained from it, but this variable will be set to the correct SKIP value when each link is printed. It is thus assumed that this variable is URL EXPORTed as well. The default is the SKIP variable named in the last <SQL> statement (in versions after Dec. 16 1999); if not given, skip is assumed.

- **NUMROWS**
  Normally pagelinks will use $rows.max to get the maximum total rows the query will return and thus compute how many links to create. If that number is known to be inaccurate, or a different limit is to be forced, the NUMROWS option can be set to the total number of rows to assume the query will return, overriding $rows.max. (Note: to just control the number of page links produced without affecting the “X through Y of Z documents” numbers, use MAXPGS.)

- **PGSZ**
  The number of rows per page. This should correspond to the SQL loop’s MAX value; in versions after Dec. 16 1999 PGSZ therefore defaults to that value. Otherwise it defaults to 10.

- **MAXPGS**

26In versions prior to Dec. 16, 1999, these were assumed to be $skip and 10, respectively.
CHAPTER 1. TEXIS WEB SCRIPT

The maximum number of page links to call the PGFUNC callback for; the default is 10. This is useful to avoid printing links for thousands of potential pages for a noisy query.

- **SUMFUNC**
  The name of the function to call to print the “X through Y of Z documents” summary. The default is sumfunc; if that function does not exist, a default summary will be printed. The SUMFUNC function has several parameters initialized when called, as described below.

- **PREVFUNC**
  The name of the function to call to print the previous-page button link. The default is prevfunc; if that does not exist a default previous-page link will be printed. See below for parameters.

- **NEXTFUNC**
  The name of the function to call to print the next-page button link. The default is nextfunc; if that does not exist a default next-page link will be printed. See below for parameters.

- **PGFUNC**
  The name of the function to call to print the numbered-page links. This is called once for each page in the projected result set, up to MAXPGS pages. The default is pgfunc; if that does not exist a default set of page links will be printed. See below for parameters.

- **ORDER**
  The order in which to call the summary, previous, next, and page link callbacks. Its value is a comma-separated list of the strings “sum”, “prev”, “next”, and/or “page”. The default is “sum,prev,next,page”: print the summary first, then the previous-page link, then the page links, then the next-page link. If a callback is not listed it will not be called.

- **URLEND**
  The additional function and MIME extension to add to URLs. This only applies to URLs printed by the builtin callbacks: if a script callback is used the programmer just prints the complete URL as needed. Normally, the builtin callbacks print URLs as $url/$urlfunc$urlext$urlq (with the original values of $urlfunc and $urlext), so that they will have the same start function as the current invocation. If the links are to enter elsewhere (but the same builtin callbacks are desired), URLEND can be set to a replacement for /$urlfunc$urlext, e.g. /search.html. URLEND was added in version 3.0.945400000 19991216.

The SUMFUNC, PREVFUNC, NEXTFUNC and PGFUNC callback functions will have the following parameters set each time they are called. Note that not all parameters are set for all callbacks:

- **startrow**
  The first row of the current page’s results, counting from 1 as $next does.

- **endrow**
  The ending row of the current page’s results, counting from 1.

- **numrows**
  The total number of rows the query will return (i.e. $rows.max or the NUMROWS value).
1.5. BUILTIN FUNCTIONS

- **pg**
  The page number (counting from 1) that this callback is to print a link for (for `PGFUNC`). This is not necessarily the current page.

- **curpg**
  The page number of the currently displayed page. The `PGFUNC` callback uses this as it iterates through pages: if $pg$ is equal to $curpg$, then a link isn’t generated because that’s the current page. (The default callback bolds this number instead of making it a link.)

- **numpgs**
  The total number of pages, counting from 1, in the entire result set.

- **pgsz**
  The number of rows per page, derived from `PGSZ` or its default.

- **valid**
  Only set for the `PREVFUNC` and `NEXTFUNC` callbacks, this is nonzero if the link is “valid”, i.e. based on the current page whether a previous or next link is valid. There’s no previous page for the first page, and no next page for the last page. Thus the default callback doesn’t print a previous/next link when this is 0.

- **prev**
  Only set for the `PREVFUNC` and `NEXTFUNC` callbacks, this is nonzero if the callback is `PREVFUNC` and zero for `NEXTFUNC`. Provides a way to combine the previous- and next-page callbacks into one function if desired.

In addition to these parameters, the parameter named by `SKIPVAR` (or its default) will be set to the proper `SKIP` value for the link being generated. Thus the variable should be `EXPORTed` to the URL for the skip to take effect on links.

**Callback Parameter Initialization**

In version 7.06.1518116000 20180208 and later, callback parameter initialization is stricter. Any required parameters (those without default values in the function declaration) that are unknown by the specific caller mechanism (`SUMFUNC` vs. `PREVFUNC` etc.) cause the callback call to fail with the error “Dynamic function call skipped: Required parameter ‘...’ for function ‘...’ cannot be provided”. This alerts the programmer to situations where a callback is expecting a parameter that cannot be provided. In addition, such unknown parameters that do have default values in the function declaration will be passed those default values – and not the same-name-as-parameter global variable value (if set), as start functions do (and callbacks used to, in earlier Texis versions). This is because unlike start functions, callbacks place a definite call with specific arguments, so arguments should only come from that call (or specified default values).

Note that the opposite case – the caller mechanism knows of parameters that are not specified in the callback function – continues to be silently supported. This allows callbacks to not specify parameters they are uninterested in, as well as to make callbacks more forward-compatible with later Texis versions that might add new parameters in caller mechanisms.
CHAPTER 1. TEXIS WEB SCRIPT

DIAGNOSTICS
pagelinks returns 0.

EXAMPLE

<EXPORT $skip URL>
<A NAME=mypg pg curpg>
<IF $pg eq $curpg> <!-- on current page already -->
   <B>$pg</B>
<ELSE>
   <!-- link to other page -->
   <A HREF=$url/search.html>$pg</A>
</IF>
</A>

<A NAME=search>
<SQL SKIP=$skip MAX=10 "select Title from books
   where Subject like $query">
   $next) $Title <P>
</SQL>
<!-- custom page links, default prev/next: -->
<pagelinks PGFUNC=mypg>
</A>

CAVEATS

The pagelinks function is available in version 3.0.943000000 19991119 and later. URLEND, and PGSZ/SKIPVAR defaults, were added in version 3.0.945400000 19991216.

The $skip or SKIPVAR variable must be set to the current page’s SKIP value when pagelinks is called. It must also be EXPORTed to the URL or query string in order for links to work, because it is modified for every callback. Any query parameters used in the <SQL> statement should be properly EXPORTed as well.

If the skip variable is local to the function that pagelinks is called from, make sure that all the callbacks declare a parameter with that same name, otherwise the skip will not be EXPORTed since it’s out of scope in the callbacks.

Note that URLEND applies only to builtin callbacks, i.e. not to script functions, where the URL is just printed as needed.

While generally consistent, the exact values printed by pagelinks can vary when called at the first row of a SQL loop vs. after the end of the loop. This depends on when $rows.min and $rows.max are known
1.5. BUILTIN FUNCTIONS

exactly: sometimes not until the last page.

The following Vortex code approximates the behavior of the built-in versions of `sumfunc`, `prevfunc`, `nextfunc` and `pgfunc`. This code can be used as a starting point for replacing one or more of these functions with custom versions:

```vortex
<EXPORT $skip $query QUERY>

<A NAME=main>
  <SQL SKIP=$skip MAX=10 ROW
    "select ... from mytable where SearchField likep $query">
    <!-- print results -->
  </SQL>
  <$DidTable = "n">
  <$Approx = "y">
  <$If $rows.min eq $rows.max><$Approx = "n"></If>
  <$SkipOrg = $skip>
  <pagelinks>
  </A>

  <$A NAME=sumfunc numrows startrow endrow pgsz skip>
  <LOCAL plural about>
  <$If $DidTable neq "y">
    <TABLE BORDER=0 WIDTH="100%" CELLPADDING=1 CELLPACING=0
      BGCOLOR="#C0C0C0">
      <$DidTable = "y">
    </TABLE>
    <$If <$numrows neq 1><$plural = "es"></IF>
    <$If $startrow gt $numrows and $Approx eq "n">
      <fmt "Past end of results (%s%kd match%s).\n"
        $about $numrows $plural>
    <$ELSE>
      <fmt "%kd through %kd of %s%kd match%s.\n"
        $startrow $endrow $about $numrows $plural>
    </ELSE>
  </If>
  </FONT></TD></TR>
  </A>

  <$A NAME=prevfunc numrows startrow pgsz>
  <$If $numrows lte $pgsz and $SkipOrg lte 0><RETURN></IF>
  <$If $DidTable neq "y">
    <TABLE BORDER=0 WIDTH="100%" CELLPADDING=1 CELLPACING=0
      BGCOLOR="#C0C0C0">
      <$DidTable = "y">
    </TABLE>
    <$If <$numrows neq 1><$plural = "es"></IF>
    <$If $startrow gt $numrows and $Approx eq "n">
      <fmt "Past end of results (%s%kd match%s).\n"
        $about $numrows $plural>
    <$ELSE>
      <fmt "%kd through %kd of %s%kd match%s.\n"
        $startrow $endrow $about $numrows $plural>
    </ELSE>
  </If>
  </FONT></TD></TR>
  </A>
```

This code can be used as a starting point for replacing one or more of these functions with custom versions.
SEE ALSO

SQL and its special variables, especially $rows.min and $rows.max
## 1.5.11 read – read files

### SYNOPSIS

```
<read $files [offset [length]]>
```

### DESCRIPTION

The `read` function reads each file given in `$files` and returns it in the corresponding value of `$ret`. If "-" (single dash) is given as a file, the standard input is read.

Normally the entire file is read. If an `$offset` is given, reading starts at that offset into the file. Negative offsets apply from the end of the file. If a `$length` is also given, that many bytes are read; the default is the rest of the file. Negative lengths indicate the rest of the file. The `$offset` and `$length` arguments are parallel to `$files`; if they have fewer values than `$files` the last value is re-used.

### DIAGNOSTICS

`read` returns a list of the contents of each file, from the indicated offset and length. Empty value(s) indicate an error reading the file or the file was empty.

### CAVEATS

The `read` function was added Sep. 17 1996. Support for reading standard input was added in version 2.6.939400000 19991008. The `$offset` and `$length` arguments were added in version 3.0.951100000 20000220.

If a file is to be immediately printed, the `spew` function is more efficient and saves memory.

### SEE ALSO

`READLN`, `spew`, `fetch`, `submit`
1.5.12 send – print raw data

SYNOPSIS

<send $values>

DESCRIPTION

send prints each of the $values out as-is, without the HTML escapement normally done with variables.

DIAGNOSTICS

send returns nothing.

CAVEATS

Full binary data is not supported yet (no ’ \0’ ASCII chars).

SEE ALSO

spew, VERB
1.5.13  **spew – print files**

**SYNOPSIS**

```
<spew $files [$offset [$length]]>
```

**DESCRIPTION**

The `spew` function reads each file given in `$files` and prints it out verbatim. If “–” (single dash) is given, the standard input is sent.

Normally the entire file is printed. If an `$offset` is given, printing starts at that offset into the file. Negative offsets apply from the end of the file. If a `$length` is also given, that many bytes are printed; the default is the rest of the file. Negative lengths indicate the rest of the file. The `$offset` and `$length` arguments are parallel to `$files`; if they have fewer values than `$files` the last value is re-used.

**DIAGNOSTICS**

`spew` returns 1 for each value of `$files` if successful, or an empty string if not.

**CAVEATS**

No path checking is done on the given files, so any file on the machine that is readable can be read, even if outside the HTML document tree.

Support for standard input was added in version 2.6.939400000 19991008. The `$offset` and `$length` parameters were added in version 3.0.951100000 20000220.

Reads are unbuffered across calls: repeated small reads of the same file are less efficient than one large read.

**SEE ALSO**

`send`, `read`, `fetch`
1.5.14 flush – flush output

SYNOPSIS

<flush>

DESCRIPTION

The flush function flushes any pending (printed but not sent) output to the server (and thus indirectly back to the web client).

Vortex buffers its output, printing a few large blocks rather than a single line at a time. This reduces the network traffic required to send the output, and can greatly increase the apparent speed of a Vortex script when viewed over a congested link.

However, in some cases this buffering can delay output, such as just before a time-consuming command. The flush function can be called beforehand to force the pending output to the server (and ultimately the user), rather than waiting for the buffer to fill.

DIAGNOSTICS

flush returns nothing.

EXAMPLE

Please wait for this command to execute: <P>
<flush>
<EXEC longcommand>
</EXEC>
<LOOP $ret>
  $ret <BR>
</LOOP>

This example flushes the output before running longcommand, which is some program that takes a while to run. This helps ensure that the “Please wait” message is viewable by the user while the command is running.

CAVEATS

The flush function was added Mar. 19 1997.
flush is only needed in exceptional circumstances. Calling it repeatedly throughout a script can needlessly increase network traffic.

It is still up to the web server to actually send Vortex’s output to the end user, and many servers buffer the output of CGI programs (like Vortex) for the same reason Vortex does. Thus, `<flush>` may not always visibly flush the output all the way to the user on some web servers. For greater control over CGI output, some servers can be configured not to buffer CGI output, or operate in “no parsed headers” (nph) mode, which may be necessary to allow a Vortex script more control over when its output is flushed (e.g. for server-push documents). Consult your web server manual for details.

SEE ALSO

fmtcp, mm, rex
1.5.15 header – print HTTP header

SYNOPSIS

<header NAME=$name [VALUE=$value] [options]>

or

<header COOKIE=$name [VALUE=$value] [options] [cookie-options]>

DESCRIPTION

The header function prints an HTTP header. Normally Vortex takes care of printing headers, such as Content-Type, before the script begins, so that all of the script’s output is part of the body. However, in some circumstances it is desirable for a script to print additional headers, override default Vortex headers, or change headers at run-time. In such cases the header function can be used.

It is important to note that since headers are only valid before the body/content of the output, the header function, if used, should be called before any content is generated. Otherwise the output of header may be suppressed, lose its effect, and/or be visible to the user (see the PRINTIF option for more info).

If the NAME option is given, a generic header will be printed. Its value will be VALUE. This form can be used to print a Location header to redirect the user, for example.

If the COOKIE option is given, a Set-Cookie header will be printed\(^{27}\). The name of the cookie will be $name, and its value will be $value, URL-encoded so that non-ASCII characters are safe. This provides a way to export “permanent” or “session” data in Vortex, since HTTP cookies are automatically URL-decoded and imported into variables (of the same name as the cookie) on future script invocations. It also provides a cross-script method of exporting data, since cookies are not inherently part of the URL.

Options valid for named headers as well as COOKIES:

- PRINTIF=$flags
  
  Only print header if conditions specified by zero or more space-separated values in $flags are met. If $flags is empty or unset, the value of [Texis] Default Header Printif (p. 578) from texis.ini is used. If that is unset, the value headers is used. A protocol and a content condition are tested; both conditions must be true for the header to be printed:

  - http (protocol)
    True if running under CGI with HTTP
  - https (protocol)
    True if running under CGI with HTTPS
  - cmdline (protocol)
    True if running from command line

\(^{27}\) A detailed discussion of HTTP cookies, their behavior and usage is beyond the scope of this manual. See RFC 2109 for more information.
1.5. BUILTIN FUNCTIONS

- headers (content)
  True if CGI and still printing headers
- content (content)
  True if command line, or CGI and past headers
- always (protocol and content)
  Always true

Multiple flags of the same type are or’ed together. The protocol condition defaults to true if no protocol flags are given. The content condition defaults to headers if no content flags are given. Added in version 5.01.1111422310 20050321.

Several additional options are valid for COOKIES only:

- DOMAIN=domain
  Specifies a domain to restrict the cookie to. Normally a browser will return the cookie only to the single host it was received from. To make the browser return it to all hosts in a given domain, e.g. x.com, set DOMAIN to “.x.com. (Note: see RFC 2109 for more on the syntax of DOMAIN.) Added in version 3.0.957500000 20000504.

- MAX-AGE=seconds
  Specifies a maximum age for the cookie, in seconds. After the given number of seconds, the user’s browser is to discard the cookie, no longer returning it in future requests. A MAX-AGE of 0 means to discard the cookie immediately. If no MAX-AGE or EXPIRES time is given, the cookie is a session cookie, and will be discarded when the browser is closed.

- EXPIRES=time
  An alternate method of specifying the maximum age of the cookie. The time is a Texis-parseable date/time, such as “+1 week”.

- PATH=path
  Gives the valid root path for the cookie. This is the subset of URLs for which the cookie is to be returned by the browser. If not specified it defaults to /, which means that all URLs for the host will be returned the cookie. A longer path could be given to restrict the cookie to a smaller URL tree.

- SECUREIF=$flags
  If conditions specified by $flags are met, print Secure flag with cookie, which instructs browsers to only return the cookie over a secure (e.g. HTTPs) connection. Syntax is the same as for PRINTIF, except that texis.ini is not checked, and the content condition defaults to false.

- HTTPONLY
  Set HttpOnly flag in header. If the browser/user-agent supports it, this flag will prevent access or modification of the cookie by client-side scripts, which may help prevent cross-site-scripting (XSS) attacks by malicious scripts. If the browser does not support the flag, it will typically be ignored. Added in version 5.01.1244880000 20090613.

Note that this is changed from previous versions, where PRINTIF implicitly defaulted to always. For backward-compatible behavior without modifying scripts, set [Texis] Default Header Printif to always in texis.ini.
Use of the \texttt{header} function is preferred over the old method of setting the URL extension to \texttt{.bin}, since it is more flexible. The script can print some, all, or none of its headers, even override Vortex headers such as \texttt{Content-Type}.

**DIAGNOSTICS**

\texttt{header} returns nothing.

**EXAMPLE**

\begin{verbatim}
<A NAME=look>
  <$pw = "FRvEhbGTRAWug">
  <pwencrypt $pass $pw>
  <IF $ret != $pw> <!-- bad password -->
    <header NAME=Location
      VALUE="http://www.example.com/login.html">
      Your login is incorrect.
      Please <A HREF=/login.html>login</A>.
    </header>
  </IF>
  <!-- Proceed with top of HTML: -->
  <BODY BGCOLOR=white>
  <H2>My Site</H2>
  </A>
\end{verbatim}

This function checks the user's password: if it is incorrect, they are redirected to a login page. This function could be called at the top of any start function to verify the user.

**CAVEATS**

The \texttt{header} function was added in version 2.6.913000000 19981207. It must be called \textit{before} any body output is printed for the header to be interpreted properly.

The \texttt{header} function controls headers printed by Vortex in CGI output. It does not control headers sent to other servers via \texttt{<fetch>}; see \texttt{<urlcp header>} (p. 198) instead.

Web servers may insert, modify, or delete headers printed by CGI programs such as Vortex, including those printed with \texttt{header}. This behavior is beyond Vortex's control. In some cases it is necessary, as with the \texttt{Location} header: the server must parse it to change the HTTP response code to 302, because the HTTP response is printed \textit{before} the \texttt{Location} header.

The \texttt{header} function does not yet work properly in conjunction with the \texttt{.bin} URL extension syntax.

Printing a \texttt{Content-Type} header may affect the default HTML-encoding of output variables, if the type is changed to/from \texttt{text/html}. If a \texttt{Content-Type} header is not printed, one will be added to the
headers as usual.

The returning of cookies on future requests, to the proper URLs, and their disposal at the \texttt{MAX-AGE} time is entirely up to the user’s browser. If the user deletes their cookie cache, the browser does not or cannot accept cookies, or has a faulty implementation, cookies may not be returned as expected.

Although \texttt{COOKIE} values are automatically URL-encoded and decoded, they are sent as plain-ASCII values, so type information is lost. Upon receipt, cookies have the same type as if they were form variables: string or integer.

\textbf{SEE ALSO}

URL syntax, \texttt{EXPORT}, RFC 2109 (cookie specification)
1.5.16 rex, split – regular expression search

SYNOPSIS

```
<rex $expr $data [ / ]>
or
<rex [ options ] $expr $data>
...
</rex>

<spllt $expr $data [ / ]>
or
<spllt [ options ] $expr $data>
...
</split>
```

DESCRIPTION

The rex function searches for each REX expression value of $expr in each value of $data. The split function acts the same way, except that it returns the non-matching data from $data (i.e. the SPLIT option below). The return type is varbyte if the $data is type varbyte or byte, otherwise it is varchar.

If given no other options, rex and split return a list of the matching (or non-matching) hits from $data. If any of the following options are given however, the statement is a looping block command, e.g. a close tag is expected, and $loop/$next are set as in SQL ($loop always starts at 0). Any statements inside the block are executed once per returned hit, with $ret being a loop variable accumulating hits. In addition, the variable $ret.off contains the integer byte offset into the current search buffer where the hit starts. The loop can be exited with BREAK or RETURN. The looping syntax was added in version 2.6.938200000 19990924; $ret.off in version 3.01.966500000 20000816.

Looping options are:

- **ROW**
  As in SQL, do not accumulate hits in $ret, and do not make it a loop variable; each new value erases the previous. ROW should be used in a looping rex/split when a large number of return values are expected but only need to be examined one at a time; this saves memory and time since all the hits do not have to be stored in memory. ROW should also be used when functions are called within the block, because otherwise $ret is a loop variable, hindering multi-value returns.

- **SKIP=$n**
  Skip the first $n hits when returning values. This does not affect the value of $loop.

- **MAX=$n**
  Return at most $n hits.

- **SPLIT**
  Instead of returning the hit data, return non-matching data, i.e. the parts of $data outside the hits. The REX expressions in effect become delimiters for the data returned. This is similar to the command-line rex option `-v` (except there are no delimiters as with command-line rex). This is the default for the split command.

- **NONEMPTY**
Ignore empty (zero-length) return values. This is useful with SPLIT when empty values are not significant.

- **SYNTAX=re2|rex**
  The $expr syntax is RE2 or REX; the default is REX. Note that the expression syntax may also be changed by prefixing the expression with "\<re2\>" or "\<rex\>". Added in version 7.06. Note that RE2 expressions are not supported on all platforms; use <vinfo features> (p. 292) or the SQL function hasFeature() to determine if RE2 is supported on the current texis -platform platform (Windows, most Linux 2.6 versions except i686-unknown-linux2.6.17-64-32 are supported). Using an RE2 expression on an unsupported platform will result in the error message “REX: RE2 not supported on this platform”.

**DIAGNOSTICS**

**rex** returns a list of the matching hits from $data. **split** returns a list of the non-matching data. If looping, the corresponding byte offset into the current search item is returned in $ret.off as well.

**REX Expression Syntax**

- Expressions are composed of characters and operators. Operators are characters with special meaning to REX. The following characters have special meaning: “\=+*?\{\}\\$.-!” and must be escaped with a “\” if they are meant to be taken literally. The string “>>” is also special and if it is to be matched, it should be written “\\>>”. Not all of these characters are special all the time; if an entire string is to be escaped so it will be interpreted literally, only the characters “\=+*\{\}\\$.-!” need be escaped.

- A “\” followed by an “R” or an “I” means to begin respecting or ignoring alphabetic case distinction. (Ignoring case is the default.) These switches stay in effect until the end of the subexpression. They do not apply to characters inside range brackets.

- A “\” followed by an “L” indicates that the characters following are to be taken literally, case-sensitive, up to the next “\L”. The purpose of this operation is to remove the special meanings from characters.

- A subexpression following “\F” (followed by) or “\P” (preceded by) can be used to root the rest of an expression to which it is tied. It means to look for the rest of the expression “as long as followed by . . .” or “as long as preceded by . . .” the subexpression following the \F or \P. Subexpressions before and including one with \P, and subexpressions after and including one with \F, will be considered excluded from the located expression itself.

- A “\” followed by one of the following C language character classes matches any character in that class: alpha, upper, lower, digit, xdigit, alnum, space, punct, print, graph, cntrl, ascii. Note that the definition of these classes may be affected by the current locale.
• A "\" followed by one of the following special characters will assume the following meaning:
  n=newline, t=tab, v=vertical tab, b=backspace, r=carriage return, f=form feed, 0=the null character.

• A "\" followed by \x{n} or \x{nn} where n is a hexadecimal digit will match that character.

• A "\" followed by any single character (not one of the above) matches that character. Escaping a character that is not a special escape is not recommended, as the expression could change meaning if the character becomes an escape in a future release.

• The character "\ˇ" placed anywhere in an expression (except after a "[" ) matches the beginning of a line (same as \x{0A} in Unix or \x{0D}\x{0A} in Windows).

• The character "$" placed anywhere in an expression matches the end of a line ( \x{0A} in Unix, \x{0D}\x{0A} in Windows).

  Note: The beginning of line ("\ˇ") and end of line ("\$") notation expressions for Windows are both identified as a 2 character notation; i.e., REX under Windows matches "\x{0D}\x{0A}" (carriage return, line feed) as beginning and end of line, rather than "\x{0A}" as beginning, and "\x{0D}" as end.

• The character "," matches any character.

• A single character not having special meaning matches that character.

• A string enclosed in brackets ("[]") is a set, and matches any single character from the string. Ranges of ASCII character codes may be abbreviated with a dash, as in "[a-z]" or "[0-9]". A "\ˇ" occurring as the first character of the set will invert the meaning of the set, i.e. any character not in the set will match instead. A literal "-" must be preceded by a "\". The case of alphabetic characters is always respected within brackets.

A double-dash ("--") may be used inside a bracketed set to subtract characters from the set; e.g. "[\alpha--x]" for all alphabetic characters except "x". The left-hand side of a set subtraction must be a range, character class, or another set subtraction. The right-hand side of a set subtraction must be a range, character class, or a single character. Set subtraction groups left-to-right. The range operator "-" has precedence over set subtraction. Set subtraction was added in version 6.

• The "\>>>" operator in the first position of a fixed expression will force REX to use that expression as the "root" expression off which the other fixed expressions are matched. This operator overrides one of the optimizers in REX. This operator can be quite handy if you are trying to match an expression with a "!" operator or if you are matching an item that is surrounded by other items. For example: "x+++y+z++" would force REX to find the "y"s first then go backwards and forwards for the leading "x"s and trailing "z"s.

• Normally, an empty expression such as "=" (i.e. 1 occurrence of nothing) is meaningless. However, if such an empty expression is the first or last in the list, and is the root expression (i.e. contains "\>>>"), it will constrain the whole expression list to only match at the start or end of the buffer. For example: ">>>first" would only match the string "first" if it occurs at the start of the search buffer. Similarly, "last>>>=" would only match "last" at the end of the buffer.

• The "!" character in the first position of an expression means that it is not to match the following fixed expression. For example: "start=!finish+" would match the word "start" and anything
past it up to (but not including the word “finish”). Usually operations involving the NOT operator involve knowing what direction the pattern is being matched in. In these cases the “>>” operator comes in handy. If the “>>” operator is used, it comes before the “!” . For example: “>>start=!finish+finish” would match anything that began with “start” and ended with “finish”. The NOT operator cannot be used by itself in an expression, or as the root expression in a compound expression.

Note that “!” expressions match a character at a time, so their repetition operators count characters, not expression-lengths as with normal expressions. E.g. “!finish{2,4}” matches 2 to 4 characters, whereas “finish{2,4}” matches 2 to 4 times the length of “finish”.

REX Expression Repetition Operators

- A regular expression may be followed by a repetition operator in order to indicate the number of times it may be repeated.
  
  Note: Under Windows the operation “\{X,Y\}” has the syntax “\{X-Y\}” because Windows will not accept the comma on a command line. Also, N occurrences of an expression implies infinite repetitions but in this program N represents the quantity 32768 which should be a more than adequate substitute in real world text.

- An expression followed by the operator “\{X,Y\}” indicates that from X to Y occurrences of the expression are to be located. This notation may take on several forms: “\{X\}” means X occurrences of the expression, “\{X,N\}” means from X to N occurrences of the expression, and “\{,Y\}” means from 0 (no occurrences) to Y occurrences of the expression.

- The “?” operator is a synonym for the operation “\{0,1\}”. Read as: “Zero or one occurrence.”

- The “*” operator is a synonym for the operation “\{0,\}”. Read as: “Zero or more occurrences.”

- The “+” operator is a synonym for the operation “\{1,\}”. Read as: “One or more occurrences.”

- The “=” operator is a synonym for the operation “\{1\}”. Read as: “One occurrence.”

RE2 Syntax

In Texis version 7.06 and later, on some platforms the search expression may be given in RE2 syntax instead of REX. RE2 is a Perl-compatible regular expression library whose syntax may be more familiar to Unix users than Texis’ REX syntax. An RE2 expression in REX is indicated by prefixing the expression with “\<re2\>” . E.g. “\<re2\>\w+” would search for one or more word characters, as “\w” means word character in RE2, but not REX.

REX syntax can also be indicated in an expression by prefixing it with “\<rex\>”. Since the default syntax is already REX, this flag is not normally needed; it is primarily useful in circumstances where the syntax has already been changed to RE2, but outside of the expression – e.g. a Vortex <rex> statement with the option syntax=rex.

Note that while the \<re2> and \<rex> escapes are supported everywhere, the RE2 expression syntax itself is not supported on all platforms. Where it is unsupported, attempting to invoke an RE2 expression
will result in the error message "REX: RE2 not supported on this platform". Use \(<\text{vxinfo features}\) (p. 292) or the SQL function \(\text{hasFeature()}\) to determine if the current \text{texis}\_platform platform supports RE2 expressions. (Windows, most Linux 2.6 versions except 1686-unknown-linux2.6.17-64-32 are supported.)

**REX Caveats and Commentary**

REX is a highly optimized pattern recognition tool that has been modeled after the \text{grep} and \text{lex} Unix family of Unix tools. Wherever possible REX’s syntax has been held consistent with these tools, but there are several major departures that may bite those who are used to using the \text{grep} family.

REX uses a combination of techniques that allow it to operate at a much faster rate than similar expression matching tools. Unlike \text{grep}, Rex is both deterministic and non-directional. This may cause some initial problems with users familiar with \text{grep}’s way of thinking.

REX always applies repetition operators to the longest preceding expression. It does this so that it can maximize the benefits of using its rapid state skipping pattern matcher.

If you were to give \text{grep} the expression: “\texttt{ab*de+}"

It would interpret it as: an “a” then 0 or more “b”s then a “d” then 1 or more “e”s.

REX will interpret this as: 0 or more occurrences of “ab” followed by 1 or more occurrences of “de”.

The second technique that provides REX with a speed advantage is ability to locate patterns both forwards and backwards indiscriminately.

Given the expression: “\texttt{abc*def}”, the pattern matcher is looking for “Zero to N occurrences of ’abc’ followed by a ‘def’”.

The following text examples would be matched by this expression:

```
    abcabcabcabcdef
    def
    abcdef
```

But consider these patterns if they were embedded within a body of text:

```
    My country ’tis of abcabcabcabcdef sweet land of def, abcdef.
```

A normal pattern matching scheme would begin looking for “\texttt{abc*}”. Since “\texttt{abc*}” is matched by every position within the text, the normal pattern matcher would plod along checking for “\texttt{abc*}” and then whether it’s there or not it would try to match “\texttt{def}”. REX examines the expression in search of the the most efficient fixed length subpattern and uses it as the root of search rather than the first subexpression. So, in the example above, REX would not begin searching for “\texttt{abc*}” until it has located a “\texttt{def}”.

There are many other techniques used in REX to improve the rate at which it searches for patterns, but these should have no effect on the way in which you specify an expression.

The three rules that will cause the most problems to experienced \text{grep} users are:
1. Repetition operators are always applied to the longest preceding fixed length expression.

2. There must be at least one subexpression that has one or more repetitions.

3. No matched subexpression will be located as part of another.

Rule 1 Example: “abc=def*” means one “abc” followed by 0 or more “def”s.

Rule 2 Example: “abc•def*” cannot be located because it matches every position within the text.

Rule 3 Example: “a+ab” is idiosyncratic because “a+” is a subpart of “ab”.

Some Useful REX Expressions

- To locate phone numbers:
  
  1?\space?(?\digit\digit\digit\digit)?[\-\space]?\digit{3}=-\digit{4}

- To locate social security numbers:
  
  \digit{3}=-\digit{2}=-\digit{4}

- To locate text between parentheses:
  
  (=[^()]++) <- without direction specification
  
  or
  
  >>=(?!)+) <- with direction specification

- To locate paragraphs delimited by an empty line and 4 spaces:
  
  >>>\n\n=\space\P\{4\}!\n\n\space\space\space\space+\F\n\n=\space\{4}

- To locate numbers in scientific notation; e.g., “-3.14 e -21”:
  
  [+\-]?\space?\space?\space?\space?\space?\space?\space?[0-9]+\.?[0-9]*\space?\space?e?\space?\space?[+\-]?\space?[0-9]+

EXAMPLE

<$data = "This is a test." "This was also a test." "So there."$>

<$expr = "[\^\alpha]\P=is" "here">$>

<rex $expr $data>

The return values in $ret would be “is” and “here”.
CAVEATS

The `rex` and `split` functions were added Oct. 14 1996. The looping syntax was added in version 2.6.938200000 19990924. At that time, both `rex` and `split` became binary compatible, i.e. `varbyte` data such as GIF images can be searched without truncation at nul values, and will return `varbyte` values. Note that `$loop` and `$next` are only set in the looping version.

SEE ALSO

`sandr`
1.5.17 `sandr` – regular expression search and replace

**SYNOPSIS**

<sandr $expr $replace $data>

**DESCRIPTION**

The `sandr` function replaces in `$data` every occurrence of a `$expr` REX expression with the corresponding string from `$replace`. It returns the `$data` list with any replacements.

If `$replace` has fewer values than `$expr`, it is “padded” with empty replacement strings for the extra search values.

**DIAGNOSTICS**

`sandr` returns the `$data` list with any replacements made.

**Replacement Strings**

- The characters “? # \{ } + \” are special. To use them literally, precede them with the escapement character “\”.
- Replacement strings may just be a literal string or they may include the “ditto” character “?”. The ditto character will copy the character in the position specified in the replace-string from the same position in the located expression.
- A decimal digit placed within curly-braces (e.g. `{5}`) will place that character of the located expression to the output.
- A “\” followed by a decimal number will place that subexpression to the output. Subexpressions are numbered starting at 1.
- The sequence “\&” will place the entire expression match (sans \P and \F portions) to the output. This escape was added in Texis version 7.06.
- A plus-character “+” will place an incrementing decimal number to the output. One purpose of this operator is to number lines.
- A “#” followed by a number will cause the numbered subexpression to be printed in hexadecimal form.
- Any character in the replace-string may be represented by the hexadecimal value of that character using the following syntax: \xhh where hh is the hexadecimal value.
EXAMPLE

<$data = "Roses are red" "Violets are blue">  
<$expr = "blue" "red">  
<$replace = "you" "dead">  
<sandr $expr $replace $data>  
<LOOP $ret>  
  $ret  
</LOOP>

The output would be:

Roses are dead  
Violets are you

CAVEATS

The sandr function was added Aug. 23 1996.

The replacements occur in the order of values in $expr, so a later $expr might match a previous $replace value.

SEE ALSO

rex, split
1.5.18 strstr, strstri – find substring in string

SYNOPSIS

<strstr $needle $haystack [$mode]>
<strstri $needle $haystack [$mode]>

DESCRIPTION

The strstr function finds the first occurrence of each $needle value in each $haystack value.

The optional $mode argument is a stringcomparemode-style (p. 120) compare mode to use; the default is the current apicp stringcomparemode, with “ignorecase” added for strstri. The $mode values are used in the same order as $haystack values. The $mode argument was added in version 6.

DIAGNOSTICS

strstr and strstri return N * H values (N and H being the number of values in $needle and $haystack), i.e., there is exactly one return value for each $needle/$haystack combination. Each return value is the character index into the $haystack value where the corresponding $needle value was first found (0 for first character), or -1 if the $needle value was not found. The first N return values are for the first $haystack value (in $needle order), the next N return values are for the second $haystack value, etc. $mode is a stringcomparemode value.

EXAMPLE

<$haystack = "This is a test." "Is that?" "My island."$>
<strstr "is" $haystack>

The return values in $ret would be 2, -1 and 3.

CAVEATS

Note that the arguments to strstr are in opposite order from the C version. The strstr and strstri functions were added Feb. 5 1997.

Note that the offsets returned are character-based, not byte-based. For byte offsets, ensure iso-8859-1 is part of $mode (e.g. pass +iso-8859-1), as character offsets in ISO-8859-1 are always the same as byte offsets – unlike the default UTF-8-based mode.

Text is compared according to apicp stringcomparemode (with “ignorecase” for strstri), or the $mode argument.
SEE ALSO

substr
1.5.19 substr – extract substring from string

SYNOPSIS

<substr $str $offset $len [$mode]>

DESCRIPTION

For every $str value, the substr function returns the substring starting at character $offset of length $len characters, for every $offset and corresponding $len value. If there are fewer values of $len than $offset, the last value of $len is re-used. Indexes start at 0 for the first character. If $offset is negative, then it applies from the end of the string instead of the start. If $len has no or negative values, the length of the rest of the string is assumed.

The optional $mode argument is a stringcomparemode-style (p. 120) compare mode to use; the default is the current apicp stringcomparemode. The $mode values are used in the same order as $len values. Currently the only pertinent $mode flag is “iso-8859-1”, which determines whether to interpret text as ISO-8859-1 or UTF-8. This can alter how many raw bytes the character-based $offset and $len arguments cover, as UTF-8 characters are variable-byte-sized. The $mode argument was added in version 6.

DIAGNOSTICS

substr returns a list of $S * $O substrings of $str ($S and $O being the number of values of $str and $offset). Uses stringcomparemode $mode.

EXAMPLE

<$off = -3 0 0>
<$len = 3 3 -1>
<substr "father" $off $len>

The return values in $ret would be “her”, “fat”, “father”.

CAVEATS

Offsets and lengths count characters, not bytes. For byte counting, set $mode to “iso-8859-1”.

SEE ALSO

strstr
1.5.20  strcmp, strcmpi – compare strings

SYNOPSIS

<strcmp $a $b [$mode]>
<strcmpi $a $b [$mode]>

DESCRIPTION

The strcmp function compares every value of $a against every value of $b as a string. For each comparison, an integer value is returned:

- $a < $b: -1
- $a = $b: 0
- $a > $b: 1

The optional $mode argument is a stringcomparemode-style (p. 120) compare mode to use; the default is the current apicp stringcomparemode, with “ignorecase” added for strcmpi. The $mode values are used in the same order as $b values. The $mode argument was added in version 6.

DIAGNOSTICS

strcmp and strcmpi return $A * $B integer values ($A and $B being the number of values of $a and $b).

EXAMPLE

<strcmpi "This" "this">

The return value in $ret would be 0.

CAVEATS

Text is compared according to apicp stringcomparemode (with “ignorecase” for strcmpi) or the $mode argument.

SEE ALSO

strncmp, strnicmp, strfoldcmp
1.5.21 strncmp, strnicmp – compare strings, fixed length

SYNOPSIS

<strncmp $a $b $len [$mode]>
<strnicmp $a $b $len [$mode]>

DESCRIPTION

strncmp and strnicmp behave like strcmp and strcmpi, respectively. However, each takes a third parameter $len which gives the maximum number of characters to compare against the corresponding value of $b. If $len has fewer values than $b, its last value is re-used; if it has no values, infinity is assumed (e.g. strcmp behavior).

The optional $mode argument is a stringcomparemode-style (p. 120) compare mode to use; the default is the current apicp stringcomparemode, with “ignorecase” added for strnicmp. The $mode values are used in the same order as $b values. The $mode argument was added in version 6.

DIAGNOSTICS

strncmp and strnicmp return $A \times B$ integer values ($A$ and $B$ being the number of values of $a$ and $b$).

EXAMPLE

<strncmp "this" "the" 2>

The return value in $ret$ would be 0.

CAVEATS

Text is compared according to apicp stringcomparemode (with “ignorecase” for strnicmp) or the $mode$ argument.

Lengths in $len$ count characters, not bytes. For byte counting, add “iso-8859-1” to $mode$.

SEE ALSO

strcmp, strcmpi
1.5.22  strlen – length of string

SYNOPSIS

<strlen $str [{$mode}]>

DESCRIPTION

strlen returns the (integer) character length of each string value of $str.

The optional $mode argument is a stringcomparemode-style (p. 120) compare mode to use; the
default is the current apicp stringcomparemode. The $mode values are used in the same order as
$str values. Currently the only pertinent $mode flag is “iso-8859-1”, which determines whether to
interpret text as ISO-8859-1 or UTF-8. This can alter how many characters long the string appears to be, as
UTF-8 characters are variable-byte-sized, whereas ISO-8859-1 characters are always mono-byte. The
$mode argument was added in version 6.

DIAGNOSTICS

strlen returns the character (not necessarily byte) length of each string value of $str, according to
stringcomparemode $mode.

EXAMPLE

<strlen 123>

The return value in $ret would be 3.

CAVEATS

The strlen function was added Feb. 5 1997.

The length returned is the number of characters (not bytes) of the string value of the argument(s), i.e.
values are cast to varchar first, and the length stops with the first nul (U+0000) character. For the full size
of varbyte values (that may contain nuls), use the varinfo function’s size action.

The number of characters returned may vary with $mode, i.e. whether “iso-8859-1” is set or not.

SEE ALSO

varinfo
1.5.23 strrev – reverse string

SYNOPSIS

<strrev $str [$mode]>

DESCRIPTION

strrev reverses the order of characters in each value of $str.

The optional $mode argument is a stringcomparemode-style (p. 120) compare mode to use; the default is the current $apicp stringcomparemode. The $mode values are used in the same order as $str values. Currently the only pertinent $mode flag is "iso-8859-1", which determines whether to interpret text as ISO-8859-1 or UTF-8: if UTF-8, the individual bytes in a contiguous UTF-8 sequence are not reversed, and ISO-8859-1 characters (or bad bytes in a UTF-8 sequence) will be re-mapped to UTF-8, which may cause the string to lengthen. The $mode argument was added in version 6.

DIAGNOSTICS

strrev returns the values of $str reversed (as character not byte strings), according to stringcomparemode $mode.

EXAMPLE

<strrev "A man, a plan, a canal - Panama!">

The return value in $ret would be “!amanaP - lanac a ,nalp a ,nam A”.

CAVEATS

The strrev function was added Feb. 5 1997.
1.5.24  upper — convert to upper case

SYNOPSIS

<upper $str [$mode]>

DESCRIPTION

The upper function converts each $str string value to all upper-case letters (non-alphabetic characters are unchanged), according to the corresponding $mode value, which is a string-folding mode in the same format as the stringcomparemode setting of apicp (p. 120). The default if empty or unspecified is the current apicp stringcomparemode setting, but upper-case.

DIAGNOSTICS

upper returns its $str parameter with all values converted to upper case characters, according to stringcomparemode $mode.

EXAMPLE

<upper "Whisper!">

The return value in $ret would be “WHISPER!”.

CAVEATS

The upper function was added Oct. 24 1996. The $mode argument, or usage of stringcomparemode, was added in version 6.

SEE ALSO

lower, strfold
1.5.25 lower – convert to lower case

SYNOPSIS

<lower $str [$mode]>

DESCRIPTION

The `lower` function converts each `$str` string value to all lower-case letters (non-alphabetic characters are unchanged), according to the corresponding `$mode` value, which is a string-folding mode in the same format as the `stringcomparemode` setting of `apicp` (p. 120). The default if empty or unspecified is the current `apicp stringcomparemode` setting, but lower-case.

DIAGNOSTICS

`lower` returns its `$str` parameter with all values converted to lower case characters. `$mode` is an optional `stringcomparemode` value.

EXAMPLE

<lower "Whisper!">

The return value in `$ret` would be “whisper!”.

CAVEATS

The `lower` function was added Oct. 24 1996. The `$mode` argument, or usage of `stringcomparemode`, was added in version 6.

SEE ALSO

`upper`, `strfold`
1.5.26 strfold – fold string by case

SYNOPSIS

<strfold $str [$mode]>

DESCRIPTION

The strfold function folds each of its $str string values, according to the corresponding $mode value. Typical usage is to fold a string to all-lower-case, or remove accents. The strfold function supports UTF-8 and all locale-independent Unicode 5.1.0 case-foldable characters.

The $mode argument is a comma-separated list of a mode and flags, in the same format as the stringcomparemode setting of apicp (p. 120). The default if empty or unspecified is the current apicp stringcomparemode setting.

DIAGNOSTICS

strfold returns its $str argument with all values folded according to the corresponding stringcomparemode $mode value.

EXAMPLE

<strfmt "%hV" "Gro&#xDF;e"> <!-- German es-zett in UTF-8 -->
<strfold $ret "+igncase"> <!-- fold mode: default+ignorecase -->

The return value in $ret would be “grosse”.

CAVEATS

The strfold function was added in version 5.01.1206664000 20080327.

SEE ALSO

strfoldcmp, upper, lower
1.5.27 strfoldcmp – compare strings folded by case

SYNOPSIS

<strfoldcmp $a $b [$mode]>

DESCRIPTION

The `strfoldcmp` function compares each string value of `$a` with the corresponding value of `$b`, returning less than zero if `$a` is less than `$b`, zero if the values are equal, or greater than zero if `$a` is greater than `$b`. The strings are compared according to the corresponding `$mode` value. The `strfoldcmp` function is more powerful than – though no necessarily compatible with – the `strcmp` and `strcmpi` functions, as it supports UTF-8 and all locale-independent Unicode 5.1.0 case-foldable characters.

The `$mode` argument is a comma-separated list of a mode and flags, with the same format and aliases as for `strfold`.

DIAGNOSTICS

`strfoldcmp` returns an integer corresponding to the sort order of `$a` and `$b` according to `stringcomparemode $mode`.

EXAMPLE

<strfoldcmp "alpha" "Bravo" "unicodemono,ignorecase">

The return value in `$ret` would be −1.

CAVEATS

The `strfoldcmp` function was added in version 5.01.1206664000 20080327.

SEE ALSO

`strfold`, `upper`, `lower`
1.5.28 sort – sort variables

SYNOPSIS

<sort [flags] $var [ICASE|NUM] [DESC|ASC] [$var2 ...]>

DESCRIPTION

The sort function takes one or more variable arguments to be sorted together. If only one variable is given, the return value is given in $ret. Otherwise (more than one variable), the variables are sorted in place, and are treated like the columns of a table to be sorted by rows. The first variable is the primary key to order by. Rows that compare the same for this variable are sorted by the next variable; if the second’s values are identical then the third is compared, etc. If all variables’ values compare equal for a given pair of rows, the original relative row order is preserved. In version 6 and later, string comparisons use the current apicp stringcomparemode setting (p. 120).

Several flags may appear after each variable, and apply only to the immediately preceding variable:

- DESC
  Sort this variable in descending order, instead of the default ascending order.

- ASC
  Sort this variable in ascending order, the default.

- ICASE
  Ignore case when sorting this variable, and sort it alphabetically if it’s non-varchar. Thus “Norway” is equal to “norway” with this flag, and the integer value 123 is less than 45.

- NUM
  Sort numerically: the variable’s values are treated as numbers (floating point). This is useful for sorting a varchar variable whose values are numbers, which would otherwise be sorted alphabetically.

In addition, there are several flags that may be given immediately before the first variable, as overall flags:

- SHORTEST
  Sort the shortest variable’s number of values. Ideally, all variables given to <sort> should have the same number of values, so that a “row” of values across the variables is never broken up. However, if one or more variables are shorter than others, this could cause a “gap” to appear in the shorter variables’ arrays (e.g. if the last row became sorted first, there would be no last-row value for the short variables to replace the first).

  With the SHORTEST flag set, these gaps are avoided by only sorting the first $N$ values, where $N$ is the number of values of the shortest variable. Later values are unsorted. Prior to version 5.01.1189552000 20070911, this was the default. In later versions, LONGEST is the default.

\(^{29}\) This is for backwards compatibility with early versions of sort.
• **LONGEST**

Sort the longest variable’s number of values. The problem of unequal length variables is handled by sorting all rows, and allowing gaps to appear in shorter variables as needed. These gaps – when they are “internal” or between pre-existing values – are populated with default/empty values. Thus, a short variable may end up with extra values after the `<sort>` (but not necessarily as many as the longest var). In version 5.01.1189552000 20070911 and later, LONGEST is the default.

If old-style SHORTEST behavior is needed as a default for some reason, it can be restored by editing texis.ini and setting [Texis] Default Vortex Sort Shortest to a non-zero value. Note that this setting is applied at compile-time only.

**DIAGNOSTICS**

`sort` returns the sorted list in `$ret` if one variable is given; otherwise it returns nothing and the variables are sorted in place.

**EXAMPLE**

```
<$colors = red orange yellow green blue violet>
<$num = "FF0000" "FFA500" "FFFF00" "00FF00" "0000FF" "EE82EE">
<sort $num DESC $colors>
<LOOP $colors $num>
  $colors $num
</LOOP>
```

The output, sorted by descending order of color values (e.g. brightness):

yellow FFFF00
orange FFA500
red FF0000
violet EE82EE
green 00FF00
blue 0000FF

**CAVEATS**

The `sort` function was added Aug. 23 1996. The SHORTEST and LONGEST flags were added in version 5.01.1189552000 20070911.

All variables given to `sort` should have the same number of values. If not, some values may be unsorted, or some variables may have additional values inserted. See the SHORTEST/LONGEST flags.

In version 6 and later, string comparisons use the current apicp stringcomparemode setting (p. 120).
This is one of the few functions in Vortex that modifies its arguments without the $\&var$ reference syntax.

SEE ALSO

uniq
1.5.29 uniq, uniqcount – produce unique list of values

SYNOPSIS

<uniq [flags] $var [ICASE|NUM] [DESC|ASC] [$var2 ...]>
<uniqcount>

DESCRIPTION

The `uniq` function behaves like `sort`, in that it sorts its arguments. It takes the same flags after variables that `sort` does. Additionally, rows that compare identical are deleted. In version 6 and later, string comparisons use the current `apicp stringcomparemode` setting (p. 120).

Several overall flags may be given before the first variable:

- **SORTED**
  Assume that the values were already sorted. Use this flag if it is known that the values are in order (as per the given flags/defaults), as it saves the time of re-sorting the variables.

- **SHORTEST**
  Sort the shortest variable’s number of values. See `<sort>` (p. 168) for details. Prior to version 5.01.1189552000 20070911, this was the default. In later versions, LONGEST is the default.

- **LONGEST**
  Sort the longest variable’s number of values. See `<sort>` (p. 168) for details. In version 5.01.1189552000 20070911 and later, LONGEST is the default.

The `uniqcount` function returns an integer list of the number of times each value/row occurred in the previous call to `uniq`.

DIAGNOSTICS

uniq returns the sorted, uniq list in `$ret` if one variable is given; otherwise it returns nothing and the variables are sorted and unique’d in place. uniqcount returns an integer list in `$ret` of the number of times each value/row occurred in the previous uniq call.

EXAMPLE

<!-- Print letter counts in the word "senselessness". -->
<$hist = s e n s e l e s s n e s s s>
<uniq $hist ICASE>
<$hist = $ret>
<uniqcount>
<LOOP $ret $hist>
  ‘$hist’ occurs $ret times
</LOOP>

The output would be:

‘e’ occurs 4 times
‘l’ occurs 1 times
‘n’ occurs 2 times
‘s’ occurs 6 times

CAVEATS

The uniq and uniqcount functions were added Aug. 23 1996.

All variables given to uniq should have the same number of values. If not, some values may be unsorted, or some variables may have additional values inserted. See the SHORTEST/LONGEST flags.

In version 6 and later, string comparisons use the current apicmp stringcomparemode setting (p. 120). Note that unlike most functions, uniq modifies its arguments.

SEE ALSO

sort
1.5.30 count – return number of variable values

SYNOPSIS

<count $var>

DESCRIPTION

The count function counts the number of values a variable has.

DIAGNOSTICS

count returns the (integer) number of values the $var variable has.

EXAMPLE

<$x = this is a test>
<count $x>
$ret values.

The output would be:

4 values.

CAVEATS

The count function was added Aug. 29 1996.
1.5.31  strtonum – convert string to number

SYNOPSIS

<strtonum $str>

DESCRIPTION

The strtonum function converts each of its string arguments to a numeric (double) value. The strings may
be any text quantity that NPM (Metamorph’s Numeric Pattern Matcher) can parse.

DIAGNOSTICS

strtonum returns the double numeric value of each of its string arguments, or 0 if unparsable.

EXAMPLE

<$txt = "one thousand nine hundred ninety eight"
    "four score and seven years ago"
    "1/3"  "3.14"  "one hundred percent"
    "two thousandths"
    "nine and a half">
<strtonum $txt>
<LOOP $ret> $ret </LOOP>

The output would be:

1998 87 0.333333 3.14 1 0.002 9.5

CAVEATS

The strtonum function was added in version 2.1.876800000 19971014.

SEE ALSO

sum
1.5.32 rand – generate pseudo-random number

SYNOPSIS

<rand $min $max>

DESCRIPTION

The `rand` function returns a pseudo-random integer \( n \) between \( \min \) and \( \max \).

DIAGNOSTICS

`rand` returns a pseudo-random integer \( n \), \( \min \leq n < \max \).

EXAMPLE

```sh
<rand 0 100>
Your test score is \$ret\%.
```

The output might be:

```
Your test score is 42%.
```

CAVEATS

The `rand` function was added Nov. 7 1996.

Although seeded, the randomness of numbers generated by `rand` may vary by platform.

SEE ALSO

`randpick, srand`
1.5.33 `randpick` – return pseudo-random value of a variable

SYNOPSIS

`<randpick $var>`

DESCRIPTION

The `randpick` function returns a single randomly-selected value of `$var`, as a string.

DIAGNOSTICS

The `randpick` function returns a single randomly-selected value of `$var`, as a string.

EXAMPLE

`<$colors = red orange yellow green blue violet>  
<randpick $colors>  
Roses are $ret.`

The output might be:

Roses are green.

CAVEATS

The `randpick` function was added Nov. 7 1996.

SEE ALSO

`rand`, `srand`
### 1.5.34 srand – seed the pseudo-random number generator

**SYNOPSIS**

<srand $seed>

**DESCRIPTION**

The `srand` function “seeds” the pseudo-random number generator with an integer `$seed`, which determines what sequence of pseudo-random numbers will be returned in future calls to `rand` and `randpick`. This can be used to help ensure the sequence is different each run (with a random seed), or to generate the same sequence on a subsequent run (with the same seed) i.e. for reproducibly random results.

If an empty string is given, a semi-random seed is created; this is also how the generator is seeded if `srand` is not called.

**DIAGNOSTICS**

The `srand` function returns the previous seed value.

**EXAMPLE**

```plaintext
<srand 12345>
<WHILE $loop lt 10>
    <rand 0 10> $ret
</WHILE>
```

The output will always produce the same sequence of “random” numbers on a given platform.

**CAVEATS**

The `srand` function was added Mar. 16 2000.

The reproducibility and/or randomness of numbers is system-dependent.

**SEE ALSO**

`rand`, `randpick`
1.5.35 exit – exit program

SYNOPSIS

<exit [$code]>

DESCRIPTION

The `exit` function causes the script to terminate immediately. It can be used to quit a script from within a complicated set of function calls, which might otherwise require numerous returns and checks. An optional exit code `$code` may be given; the default is 0.

DIAGNOSTICS

`exit` returns nothing.

EXAMPLE

```
<A NAME=checklogin>
  <$pw = "FRvEhbGTRAWug">
  <pwencrypt $pass $pw>
  <IF $ret neq $pw>
    Your password is incorrect.
    <exit>
  </IF>
</A>
```

This function could be called to check a user’s password at any point, without having an `IF` statement after every call: it only returns if the password is correct.

CAVEATS

State information may be lost if `$url` hasn’t been printed yet.

SEE ALSO

RETURN
1.5. BUILTIN FUNCTIONS

1.5.36 fetch – fetch URLs

SYNOPSIS

<fetch [METHOD=$methods] [RENAMETO=$path] $theurl [$rawdoc]>

or

<fetch PARALLEL[=n] [METHOD=$methods] [RENAMETO=$path] [URLS=]$urls
   [$loopvar ...]>

...</fetch>

DESCRIPTION

The fetch function retrieves URLs, using the HTTP GET method (or its equivalent for other protocols), and returns the (unformatted) documents in $ret. The METHOD option, added in version 3.01.9713000000 20001011, specifies a parallel list of alternate method(s); each method may be one of OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, MKDIR, or RENAME. Not all methods are supported by all protocols. Some methods are mapped to an equivalent method when used with non-HTTP protocols. If the RENAME method is set (valid only for ftp URLs), the RENAME option must also be set to the target file path.

With the first syntax, a single URL (the first value of $theurl) is retrieved. The urlinfo function (p. 185) can then be called to obtain more information about the page just fetched. If a second argument ($rawdoc) is given, it is used as the raw document source, instead of actually fetching the URL. This provides a way to obtain the text, links, etc. of arbitrary script-generated HTML that isn’t present on an actual web page. (To just HTML-decode an arbitrary string without tag removal, word-wrapping or link parsing, see the %H code to strfmt with the ! flag, p. 101.)

With the second (loop) syntax, all of the URLs in $urls are fetched in parallel, that is, simultaneously. Once the first (i.e. quickest) URL is completed, the fetch loop is entered, with $ret set to the raw document source. As subsequent URLs are completed, the loop is iterated again; once for each member of $urls. Inside the loop, the urlinfo function can be used to retrieve further information about the URL just completed.

It is important to note with the loop syntax that URLs are returned fastest-first, which might not be the order they are present in $urls. For example, suppose two URLs are being fetched where the first URL takes 10 seconds to download and the other 3 seconds. With the parallel loop syntax, the second will probably be returned first, after 3 seconds; then 7 seconds later the first will be completed. A URL that refers to an unresponsive web server will not hold up other URLs; it is merely returned last, when it times out.

As an aid in tracking which URL was returned in each iteration, the $urls variable and any subsequent $loopvar variables are looped over in the fetch, but in the same order as returned URLs. Thus $urls

30Note that not all fetch-related processing may occur when the $rawdoc argument is provided, e.g. content codings are not currently decoded.
is set to the URL just retrieved inside the loop.

The special variables $\text{loop}$ and $\text{next}$ are set and incremented inside the loop as well: $\text{loop}$ starts at 0, $\text{next}$ at 1.

If an argument to \texttt{PARALLEL} is given, only that many URLs will be fetched simultaneously; the remaining ones are started only as the first ones complete. The default (no argument to \texttt{PARALLEL}) is to start fetching all URLs initially (in version 4 and earlier) or only 10 at a time (version 5 and later).

\textbf{DIAGNOSTICS}

\texttt{fetch} returns the raw document just fetched (after content/transfer encodings decoded), or the value of $\text{rawdoc}$.

\textbf{EXAMPLE}

This example uses the loop syntax to search multiple search engines simultaneously. First, the user’s query is placed into each URL with \texttt{sandr}. Then the resulting URLs are fetched; because of the \texttt{PARALLEL} flag, the fastest engine will return first. Each page is then post-processed to remove HTML outside the \texttt{<BODY>}</BODY>} tags – since there will be multiple pages concatenated together – and displayed following a \texttt{<BASE>} tag so that the user’s browser knows the proper path for the links:

```text
$urlcp$timeout 10>
<$rooturls =
    "http://www.searchsite1.com/cgi-bin/search?q=@@@"
    "http://www.searchsite2.com/findit?query=@@@"
    "http://www.searchsite3.com/cgi-bin/whereis?q=@@@&cmd=search" >
<strfmt "%U" $query> <!-- URL-escape query -->
<sandr "[!\?\#\{\}\\]+" "\\1" $ret> <!-- make sandr-safer -->
<sandr "@@@" $ret $rooturls> <!-- and insert into URLs -->
<$urls = $ret>
$BODY BGCOLOR=white>
<fetch PARALLEL $urls>
    <$html = $ret>
    <$urlinfo actualurl>
    <$BASE HREF="$ret"> <!-- tell browser the base -->
    <$send $html> <!-- print site results -->
</fetch>
</BODY>
```

\textbf{CAVEATS}
The PARALLEL syntax to fetch was added in version 2.1.902500000 19980807. Support for FTP was added in June 1998, Gopher in version 2.6.938200000 19990924, HTTPS and javascript: June 17 2002, and file:// URLs in version 4.02.1048785541 20030327. Protected file:// URLs (requiring user/pass) supported for Windows in version 5.01.1123012366 20050802. RENAMETO was added in version 5.01.1173117000 20070305.

All URLs are returned, even those that cause an error (empty string returned). The urlinfo function can then be used to obtain the error code.

In versions prior to 3.0.949000000 20000127, or if <urlcp dnsmode sys> is set, domain name resolution cannot be parallelized due to C lib constraints. Thus, an unresponsive name server (as opposed to a web server) may hold up other URLs, or even exceed the <urlcp timeout> setting. In some versions, parallel FTP retrieval is not supported.

Note that $loop and $next are merely incremented inside the loop: they do not necessarily correspond to the array index of the currently returned URL.

As little work as possible should occur inside a fetch loop, as any time-consuming commands could cause in-progress fetches to time out.

SEE ALSO

submit, urlinfo, urlcp
1.5.37 submit – submit HTML form

SYNOPSIS

<submit [options] URL=$theURL [var=value ...]>

DESCRIPTION

The submit function submits an HTML form, i.e. as a Web browser would. The given URL is fetched, using the HTTP POST method (or its equivalent for non-HTTP URLs). The submission will include the (optional) list of variables given, URL encoded. Multi-valued variables will have each of their values sent as a separate value, e.g. for a multi-value SELECT box on a form.

Options that may be set before the variables are:

- **URL=$theURL**
  The URL to fetch; required. Note that if it contains a query string it will be replaced by the given variables list if the METHOD is GET or HEAD.

- **METHOD=$method**
  The HTTP method to use (or its equivalent if a non-HTTP URL is given). The default is POST. The $method may be one of OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, MKDIR or RENAME. Note that not all web servers support all methods; most implement only GET, HEAD and POST reliably. Consult an HTTP reference, such as RFC 2068, for details on these methods. Note that for FTP URLs, only GET, PUT and DELETE are supported. For Gopher URLs, the method is irrelevant. Other protocols may not support all methods, or may map some methods to an equivalent action.

- **DATA=$data**
  The raw data to send. This is generally required for FTP URLs with the PUT method, and in conjunction with CONTENT-TYPE is useful in specifying an alternate content for HTTP submissions. Note that content is only sent for the methods POST, PUT, DELETE and OPTIONS. Supersedes FROMFILE and the variable list, if any. Added in version 2.1.900648000 19980717.

- **FROMFILE=$file**
  The name of the file to send as content. This is useful when sending a large file via FTP, which may not fit in memory if read separately and sent as a DATA buffer. If the $file is “-” (dash), the standard input is sent instead; this is only valid for non-CGI (command-line) scripts. The FROMFILE option was added in version 2.6.938460000 19990927.

- **TOFILE=$file**
  The name of the file to save the content to. This is useful when receiving a large file via FTP which may not fit in memory as $ret. If the $file is “-” (dash), the content is written to the standard output of the script. Note that setting TOFILE will cause the usual output document returned via $ret to be empty, and any output-derived fields obtainable via urlinfo such as links, formatted document, etc. will be unavailable. Added in version 2.6.938460000 19990927.
1.5. BUILTIN FUNCTIONS

- CONTENT-TYPE=$type
  Specify the content-type of the data given with DATA. The default is none, or "application/x-www-form-urlencoded" if variables are given directly. Added in version 2.1.900648000 19980717.

DIAGNOSTICS

submit returns the raw document fetched.

EXAMPLE

The following HTML form found on a web site:

```html
<FORM METHOD=post ACTION=http://www.xyz.com/cgi-bin/prog>
  Name:  <INPUT NAME=Name>
  Colors: <INPUT TYPE=checkbox NAME=Color VALUE=red>red
          <INPUT TYPE=checkbox NAME=Color VALUE=blue>blue
          <INPUT TYPE=checkbox NAME=Color VALUE=green>green
  <INPUT TYPE=submit>
</FORM>
```

could be submitted to in Vortex as follows:

```vortex
<$mycolors = "red" "green">
<submit METHOD=POST URL=http://www.xyz.com/cgi-bin/prog
  Name="Joe Bob"
  Color=$mycolors>
```

CAVEATS

The submit function was added Sep. 17 1996. Support for FTP was added in June 1998, and Gopher in version 2.6.938200000 19990924.

In versions prior to 3.0.949000000 20000127, or if <urlcp dnsmode sys> is set, domain name resolution may exceed the <urlcp timeout> setting (due to C lib constraints).

Any query string already present in the given URL will necessarily be stripped if the METHOD is GET and one or more variables are given.

If TOFILE is given, the normal output returned via $ret is empty, and content-derived fields such as links, formatted text, etc. obtainable with urlinfo are unavailable. This is because the (potentially large) data being received is sent directly to a file without processing.
SEE ALSO

fetch, urlinfo, urlcp
1.5. BUILTIN FUNCTIONS

1.5.38 urlinfo – get detailed page info

SYNOPSIS

<urlinfo $name [$which]>

DESCRIPTION

The urlinfo function returns information about the last page retrieved with fetch or submit. Inside a fetch loop (e.g. with the PARALLEL flag), this is the page just returned for the current loop. The $name argument describes what to return; some values take a second $which argument (as noted below). Possible values for $name and what they return are:

- **actualurl (string)**
  The last URL retrieved. It may differ from the argument to fetch or submit, e.g. if redirects were followed. See also intermediateurls (p. 189).

- **allrefs (list)**
  The list of all links, images, frames, iframes and string links from the document; essentially a concatenation of the links, images, frames, iframes and strlinks values, but with duplicate entries from the same tag/attribute tuple removed. Added in version 7.06.1.1463100000 20160512. Note that the urlcp settings getframes, getiframes and/or getscripts may affect which URLs are returned.

- **authparams (list)**
  The list of names of parsed authentication parameters sent by the server. The value for a particular parameter name can be obtained with authparam $param. Added in version 5.1. Note that authentication parameters may not be available even if authentication is used, if the server does not send them. For example, the second and later requests on a connection may not need parameters, if credentials are sent with the initial request and thus the server does not need to challenge in the response.

- **authparam $param (string)**
  The authentication parameter $param from the server. $param may be realm for the Basic authentication realm, target for the NTLM target (i.e. domain), or serverchallenge for the NTLM server challenge nonce. Added in version 5.1. Authentication parameter names are case-insensitive.

- **authscheme (string)**
  The authentication scheme used. Returns one of the scheme tokens used by <urlcp authschemes> (p. 208). Added in version 5.01.1.1239140000 20090407.

- **authschemes (list)**
  The list of authentication schemes currently allowed via <urlcp authschemes> (p. 208). Added in version 7.04.
• authschemehighest (string)
The highest (most secure) authentication scheme used during the entire transaction, i.e. across
redirects (if any). E.g. if a Basic authentication protected page was fetched, which then redirected
to an anonymous-access page, authschemehighest would return Basic, even though
authscheme would return anonymous (from the last page). Returns one of the scheme tokens
used by <urlcp authschemes> (p. 208). Added in version 5.01.1239140000 20090407.

• charsetconfigtotext (string)
The current charset configuration, in the format used by <urlcp charsetconfigfromfile>

• charsetdetected (string)
The charset of the source page, as detected by scanning the document, without parsing explicit
charset labels. Added in version 5.

• charsetexplicit (string)
The charset of the source page, as explicitly set in a header or <META HTTP-EQUIV> label.
Returns Unknown if unknown or not set. Added in version 5.

• charsetsrc or charsetsource (string)
The charset of the source page, as interpreted by the parser. This is taken from the first available
source, in descending priority: the charset as set by <urlcp charsetsrc>; the charset explicitly
set in the page (header or meta); the charset detected by scanning the document; or the <urlcp
charsetsrcdefault> charset. Added in version 5.

• charsetttxt or charsettext (string)
The charset of the formatted text (as returned by <urlinfo text>). Added in version 5.

• contenttype (string)
The MIME content type of the page (without any parameters). This may have been derived from the
Content-Type header, a <META HTTP-EQUIV> tag, or the URL extension, depending on what
is available. In version 7.06.1477065000 20161021 and later, the value is returned lower-case for
easier comparison, since media types are case-insensitive.

• contenttypeparams (list)
The names of parameters in the MIME content type, if any. In version 7.06.1477065000 20161021
and later, the names are returned lower-case for easier comparison, since media types parameter
names are case-insensitive.

• contenttypeparam (list, 2 args)
The value(s) of the content type parameter(s) named $which. Multiple values may be given in
$which. Parameter names are case-insensitive.

• contenttypesrc (string)
Returns the source of contenttype and related data, i.e. how it was determined. One of
“generated”, “header”, “doctype”, “metaheader”, “urlpath”, “contentscan” or
“unknown”. Added in version 5.01.116341784 20050517. Aka contenttypesource.

• cookiejar [all] [netscape4x]
The contents of the “cookie jar” (Vortex’s internal cache of cookies received or set). Returned as a Netscape-cookie-file format text buffer. By default, only persistent (non-session) cookies are returned, i.e. the ones to be preserved across browser invocations. If the argument all is given, all cookies, including session cookies, are returned. Added in version 4.01.1022000000 20020521.

In version 5.01.1244880000 20090613 and later, a new fifth column was inserted in the output, containing the IsHttpOnly boolean value. To obtain the Netscape-4.x-compatible format of prior versions, set the netscape4x flag. <urlcp cookiejar> will accept input in either format.

- **domvalue $dom**
  Gets the value of the DOM item indicated by $dom. Note that this is not the JavaScript DOM, but the near-parallel page DOM. This can be used to get the submit URL and content for a form on the page just fetched, e.g. `document.forms.myForm.submitUrl` and `document.forms.myForm.submitContent`, after optionally setting form input values via `<urlcp domvalue>`. Added in version 5.

- **downloaddoc (string or varbyte)**
  The network-transferred downloaded document body. This is the same as rawdoc if the document had no content/transfer encodings. If it did have encodings, this is the chunked/compressed/etc. document, before decompression into rawdoc. The downloaded document is normally discarded if different from rawdoc to save memory; thus it may be empty for documents with encodings. Set `<urlcp savedownloaddoc on>` (normally off) to preserve the downloaded document (at potential cost in memory). Added in version 5.01.1249203000 20090802. See also rawdoc, which is usually more useful.

- **encodings (list)**
  The list of content/transfer encodings of the response document, in the order they were applied by the server. Known encodings (e.g. gzip) are canonicalized and lowercase. Note that known and enabled encodings are already decoded (in reverse order) in the `<fetch>` or `<urlinfo rawdoc>` returned document. Added in version 5.01.1249203000 20090802.

- **frames (list)**
  The list of frame URLs in the document. If the urlcp setting getframes is true, the list is empty since the frames have been fetched and appended to the document.

- **iframes (list)**
  The list of `<IFRAME>` URLs in the document. If the urlcp setting getiframes is true, the list is empty since the iframes have been fetched and inserted into the document.

- **headers (list)**
  The names of HTTP headers received with the document.

- **header $hdrName (list)**
  The full value(s) of the HTTP header(s) with single name $hdrName. Header names are case-insensitive.

- **headervalue $hdrName**
  The leading value (i.e. before the “;”) of the header(s) with single name $hdrName, where the header is in semicolon-parameterized format, i.e.:
value; param1=val1; param2="val 2"; ...

Added in version 6.00.1287436000 20101018.

- **headerparams** $hdrName
  The parameter name(s) from the semicolon-parameterized header(s) with single name $hdrName. Added in version 6.00.1287436000 20101018.

- **headerparam** $hdrName $paramName
  The parameter value(s) of the parameter(s) with single name $paramName from the semicolon-parameterized header(s) with single name $hdrName. Added in version 6.00.1287436000 20101018.

- **errnum** (integer)
  The Vortex fetch error code (*not* the HTTP or other protocol code), indicating a problem with the fetch. This can be non-zero even for a partially successful fetch, e.g. 15 if the page is too big. 0 indicates a completely successful fetch. See p. 194 for a list of errnum codes and what they mean.

- **errtoken** (string)
  A string token representing the numeric errnum code, e.g. DocNotFound for error 24 (Document not found). This can be used in scripts as a more readable and self-documenting value than errnum integer values, and more constant than errmsg values (which may change in future releases). See p. 194 for a list of tokens and corresponding numbers and meanings. Added in version 5.01.1246963000 20090707.

- **errmsg** (string)
  A human-readable string description of the errnum code. See p. 194 for a list of possible error messages and numbers.

- **httpcode** (integer)
  The value of the protocol response code, if any (for HTTP or FTP). Note that this varies depending on the fetched URL protocol; the errnum value is more consistent. Typical HTTP codes and what they mean are listed below. Note that this is not an exhaustive list, as the protocol code is created and sent by the web server, not Vortex. Codes will also vary for other (non-HTTP) protocols, e.g. FTP:

  - 200 Ok (all 2.N.N codes)
  - 201 Created
  - 202 Accepted
  - 204 No Content
  - 300 Redirect (all 3.N.N codes)
  - 301 Moved permanently
  - 302 Moved temporarily
  - 303 See Other
  - 304 Not modified
  - 400 Bad client request (all 4.N.N)


1.5. BUILTIN FUNCTIONS

- 401 Unauthorized
- 403 Forbidden
- 404 Not found
- 405 Method not allowed
- 406 Method not acceptable
- 407 Proxy access unauthorized
- 408 Request timed out
- 413 Request entity too large
- 414 Request URI too large
- 500 Internal server error (all 5.N.N)
- 501 Not implemented by server
- 502 Bad gateway
- 503 Service unavailable

- `httpmsg (string)`
  The protocol response string, if any (HTTP or FTP). Varies by protocol and server; check `errmsg` instead for more portable (platform-independent) messages.

- `images (list)`
  The list of image URLs in the document, e.g. `<IMG>` tags, background images, etc.

- `intermediateurls (list)`
  The list of intermediate URLs, if any, that were fetched before the final URL returned. This includes redirects, FTP/file dir/file retries, authorization retries, OPTIONS Upgrade, CONNECT tunnels, and proxy retries. Added in version 7.05.1450220000 20151215. See also `actualurl` (p. 185).

- `links (list)`
  The list of non-image link URLs in the document, e.g. `<A HREF>` tags, `<FORM>` tags, etc. Same as the return value of the obsolescent `urllinks` function. Note that frames will be listed as links if the `urlcp` setting `getframes` is false, iframes will be listed if `getiframes` is false, and script sources will be listed if `getscripts` is false. Note also that JavaScript string links (p. 193) are not included in this list, as they are unreliable; but ordinary JavaScript links are included.

- `metaheaders (list)`
  The names of `<META HTTP-EQUIV>` tags in the document.

- `metaheader $hdrName (list)`
  The entire value(s) of the `<META HTTP-EQUIV>` tag(s) with single name `$hdrName`. Header names are case-insensitive.

- `metaheadervalue $hdrName`
  The leading value (i.e. before the “;”) of the meta header(s) with single name `$hdrName`, where the header is in semicolon-parameterized format (see `headervalue` p. 187 for format example). Added in version 6.00.1287436000 20101018.
• metaheaderparams $hdrName
  The parameter name(s) from the semicolon-parameter-format meta header(s) with single name $hdrName (see headervalue p. 187 for format example). Added in version 6.00.1287436000 20101018.

• metaheaderparam $hdrName $paramName
  The parameter value(s) of the parameter(s) with single name $paramName from the semicolon-parameter-format meta header(s) with single name $hdrName (see headervalue p. 187 for format example). Added in version 6.00.1287436000 20101018.

• metanames (list)
  The names of <META NAME> tags in the document.

• metaname $hdrName (list)
  The entire value(s) of the <META NAME> tag(s) with single name $hdrName. Names are case-insensitive.

• metanamevalue $hdrName
  The leading value (i.e. before the “;”) of the meta name tag(s) with single name $hdrName, where the tag is in semicolon-parameter format (see headervalue p. 187 for format example). Added in version 6.00.1287436000 20101018.

• metanamerecs $hdrName
  The parameter name(s) from the semicolon-parameter-format meta name tag(s) with single name $hdrName (see headervalue p. 187 for format example). Added in version 6.00.1287436000 20101018.

• metanamerecparam $hdrName $paramName
  The parameter value(s) of the parameter(s) with single name $paramName from the semicolon-parameter-format meta name tag(s) with single name $hdrName (see headervalue p. 187 for format example). Added in version 6.00.1287436000 20101018.

• originalurl (string)
  The original URL retrieved (i.e. the one given to fetch or submit). It may differ from the actual last URL retrieved, e.g. if redirects were followed. Added in version 5.01.1205285000 20080311.

• prngdpid [ $path ] (integer, 2 args)
  The process ID of the prngd daemon (entropy gatherer) running on Unix file pipe $path, 0 if none detected, -1 on error. If no $path (or an empty one) is given, all standard paths (“/var/run/egd-pool”, “/dev/egd-pool”, “/etc/egd-pool”, “/etc/entropy”) and the configured path ([Texis] Entropy Pipe value in texis.ini) are checked. The prngd daemon is used on certain Unix platforms (those without /dev/random) to provide entropy to seed the random number generator for the SSL/HTTPS plugin. The prngdpid value provides a way to check if the daemon is running. Note that not all platforms require an entropy daemon. Added in version 4.01.1031761163 20020911. See also the entropypipe setting of urlcp (p. 212).

• putmsgs (list)
  The fetch-related putmsgs since the most recent <fetch> or <submit>. When called inside a <fetch parallel> loop, only the messages from the just-completed fetch are returned, making disambiguation much easier than with the standard <putmsg> function callback mechanism. If
<urlcp putmsg save> is off (p. 229), no messages will be saved or returned. The message buffer is cleared at the start of each <fetch> or <submit>. If parsing these messages, it may be helpful to turn off <urlcp putmsg pass>, so that the same messages need not be seen and parsed by the script-wide <putmsg> function callback. Added in version 6.

- **processedchunks (strings or varbyte values)**
  The ordered list of HTML document chunks that were actually processed during HTML parsing. The concatenation of these is normally the same as rawdoc. However, the chunks may differ if rawdoc is not UTF-8, as the chunks always are. The chunks may also differ from rawdoc if JavaScript was run and modified the document; e.g. some of the chunks may be the output of document.write() statements, whereas rawdoc is always the static original document. The chunks may be zero-length/empty if no HTML processing was done, e.g. for an image. Added in version 6. The concatenation of processedchunks is available as processeddoc, which may be easier to use if individual chunks (e.g. static vs. dynamic content) are not needed.

- **processedchunksbufnums (list of integers)**
  The ordered list of buffer numbers that the corresponding processedchunks values come from. During HTML and JavaScript processing, a document will end up with one or more buffers, the first of which (buffer 0) is the original static document source itself. JavaScript processing may create further buffers (e.g. the output of document.write()). A buffer may end up split into multiple chunks for HTML formatting if such JavaScript output occurs mid-buffer. For example, a document.write() in the middle of an HTML page may result in 3 chunks: the first part of buffer 0 (static doc), all of buffer 1 (generated by JavaScript), and the latter part of buffer 0 (rest of static doc). Added in version 6.

- **processeddoc (string or varbyte)**
  The concatenation of processedchunks. Added in version 7.06.1463504000 20160517.

- **rawdoc (string or varbyte)**
  The document source (after any content/transfer encodings are decoded). Same as the return value of the original fetch or submit. See also downloaddoc.

- **redirs (integer)**
  The number of redirects encountered.

- **saslmechanisms (list)**
  The list of enabled SASL mechanisms (under Negotiate authentication). See <urlcp saslmechanisms> (p. 210) for more info. A putmsg is generated if SASL is not supported on the current platform.

- **saslmechanismsavailable (list)**
  The list of available SASL mechanisms. See <urlcp saslmechanisms> (p. 210) for more info. A putmsg is generated if SASL is not supported on the current platform.

- **saslpluginpath (string)**
  The colon-separated path to look for SASL plugins in. See <urlcp saslpluginpath> (p. 210) for more info. A putmsg is generated if SASL is not supported on the current platform.

- **secure (list)**
Which parts of the transaction were conducted securely (via SSL). One or more of the following values:

- **request** - The final URL request to the server was secure.
- **response** - The final response from the server was secure.
- **ancestors** - All previous requests and responses that led to the final fetch (i.e. earlier redirects) were secure.
- **descendants** - All requests and responses made to sub-objects on the final page (e.g. frames, scripts) were secure.
- **all** - All requests and responses for the entire transaction – ancestors (if any), final page, and descendants (if any) – were secure.

Added in version 5.01.1184803500 20070718. Note that the definition of “secure” for this option only applies to the first-hop network connection (Vortex); if a proxy is used, the transaction(s) from the proxy to the URLs may or may not be secure. See also the **insecure** option.

• **insecure** (list)
Which parts of the transaction were **insecure**, i.e. not conducted securely via SSL. One or more of the following values:

- **request** - The final URL request to the server was insecure.
- **response** - The final response from the server was insecure.
- **ancestors** - One or more previous requests or responses that led to the final fetch (i.e. earlier redirects) were insecure.
- **descendants** - One or more requests or responses made to sub-objects on the final page (e.g. frames, scripts) were insecure.
- **all** - The request and response for the final page were insecure, one or more ancestors (if any) were insecure, and one or more descendants (if any) were insecure.

Added in version 5.01.1184803500 20070718. Note that the definition of “insecure” for this option only applies to the first-hop network connection (Vortex); if a proxy is used, the transaction(s) from the proxy to the URLs may or may not be secure. See also the **secure** option.

• **sslciphers** [$group] (string)
Returns the list of SSL ciphers currently set with `<urlcp sslciphers>` (p. 216), or empty string if none set (i.e. the OpenSSL default list is in effect). Added in version 7.03.1436205000 20150706.

In version 7.07 and later, an optional cipher $group may be given, to return the cipher list for that protocol group. The group may be **SSL** (the default) for protocols TLSv1.2 and below, or **TLSv1.3** for TLSv1.3 ciphers; the two lists are independent.

• **sslservercertificate** (PEM string)
Returns the SSL certificate obtained from the server, in PEM format, or empty if none (e.g. no HTTPS/SSL server contacted). If the server is an Apache or Texis Monitor web server, this certificate is typically from the server's **SSLCertificateFile** setting. The **urlutil** action **sslcertificate** (p. 238) may be used to decode the certificate into a human-readable string format. Note that a server certificate may sometimes be obtainable from an HTTPS/SSL server even if the connection fails (e.g. due to verification problems). Added in version 6.00.1320460000 20111104.
1.5. BUILTIN FUNCTIONS

- **sslclientcalist (list)**
  Returns the list of CA (certificate authority) certificate names that the HTTPS/SSL server requested as acceptable issuers of the client’s certificate. (If the server is an Apache or Texis Monitor web server, this list is typically from the server’s SSLCARequestFile or SSLCACertificateFile setting.) This is a list of certificate issuers that the server indicates it will accept as signers of the client’s (Vortex fetch lib’s) certificate. In other words, the certificate set with `<urlcp sslcertificatefile>` (p. 213) should have been signed by one of these issuers, or the server might reject the connection with a “Cannot complete SSL handshake: ... alert bad certificate” (or “... alert unknown ca”) or similar error.

  If an HTTPS/SSL server was not contacted, or the server did not request a client (Vortex) certificate for verification, this list may be empty. Added in version 6.00.1320460000 20111104.

- **sslverifyservererrtoken (string)**
  The string token that identifies the reason for the `<urlcp sslverifyserver>` error, i.e. the token for the reason part of the “Cannot verify certificate from host:port: reason at depth N” message. If no server-certificate verification was performed (e.g. sslverifyserver is off, or no SSL server was contacted), the token is empty or “unknown”. If verification was performed successfully (no errors), “Ok” is returned.

  To continue to verify SSL server certificates – but ignore this particular sub-type of verification error – this error can be disabled by adding the token prepended with a “-” (minus sign) to the `<urlcp sslverifyserver>` (p. 215) setting. Added in version 6.00.1320460000 20111104. The list of possible tokens is detailed in the SSL Client/Server Certificate Verification appendix, p. 605. Note that disabling individual sslverifyserver errors should be done with caution, as it can weaken the security provided by those checks.

- **strlinks (list)**
  The list of JavaScript string links. These may be unreliable or require further processing, so they are not returned as part of the normal links list. See also `<urlcp scriptstrlinks>` (p. 222). Added in version 5.00.1086804521 20040609.

- **sspipackages (list)**
  The list of enabled SSPI packages enabled/offered under Negotiate authentication. See `<urlcp ss pipackages>` (p. 210) for more info. A putmsg is generated if SSPI is not supported on the current platform (e.g. non-Windows).

- **sspipackagesavailable (list)**
  The list of available SSPI packages. See `<urlcp ss pipackages>` (p. 210) for more info. A putmsg is generated if SSPI is not supported on the current platform (e.g. non-Windows).

- **strbaseurls (list)**
  The list of JavaScript base URLs corresponding to strlinks. If `<urlcp scriptstrlinksabs>` is off, this enables the strlinks list to be made absolute, perhaps after some post-processing. Added in version 5.00.1086804521 20040609.

- **text (string)**
  The formatted text of the document. Same as the return value of the obsolescent `urltext` function.

- **textformatter (string)**
A token describing what formatter was used to produce the `<urlinfo text>` value; one of the following:

- **unknown** Formatter is unknown.
- **rawdoc** No formatting: text is the raw document source.
- **text** Plain-text document formatter.
- **gopher** Gopher menu formatter.
- **html** HTML document formatter.
- **rss** RSS feed formatter.
- **frame** Framed document formatter/aggregator.

Added in version 5.01.1257475000 20091105. `rss` was added in version 7.02.1407881000 20140812.

- **title** (string)
  The formatted title text of the document.

- **time or totaltime** (double)
  The total time in seconds (including fraction) to retrieve the page. This includes DNS resolution plus content transfer time. Added in version 3.01.966019604 20000811.

- **dnstime** (double)
  The time in seconds (including fraction) to resolve the hostname(s) via DNS. Added in version 3.01.966019604 20000811.

- **transfertime** (double)
  The time in seconds (including fraction) to transfer content to/from the web server. This is a more accurate measure of web server throughput because it does not include the time to resolve the hostname(s). Added in version 3.01.966019604 20000811.
The possible `errno`, `errno` and `errmsg` values are:

<table>
<thead>
<tr>
<th>errtoken</th>
<th>errmsg</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Ok Ok</td>
</tr>
<tr>
<td>1</td>
<td>ClientErr Unknown client error</td>
</tr>
<tr>
<td>2</td>
<td>ServerErr Server error</td>
</tr>
<tr>
<td>3</td>
<td>UnkResponseCode Unrecognized response code</td>
</tr>
<tr>
<td>4</td>
<td>UnkProtocolVersion Unrecognized protocol version</td>
</tr>
<tr>
<td>5</td>
<td>ConnTimeout Connection timeout</td>
</tr>
<tr>
<td>6</td>
<td>UnkHost Unknown host</td>
</tr>
<tr>
<td>7</td>
<td>CannotConn Cannot connect to host</td>
</tr>
<tr>
<td>8</td>
<td>NotConn Not connected</td>
</tr>
<tr>
<td>9</td>
<td>CannotCloseConn Cannot close connection</td>
</tr>
<tr>
<td>10</td>
<td>CannotWriteConn Cannot write to connection</td>
</tr>
<tr>
<td>11</td>
<td>CannotReadConn Cannot read from connection</td>
</tr>
<tr>
<td>12</td>
<td>CannotWriteFile Cannot write to file</td>
</tr>
<tr>
<td>13</td>
<td>OutOfMem Out of memory</td>
</tr>
<tr>
<td>14</td>
<td>PageTrunc Page not expected size, possibly truncated</td>
</tr>
<tr>
<td>15</td>
<td>MaxPageSizeExceeded Max page size exceeded, truncated</td>
</tr>
<tr>
<td>16</td>
<td>TooManyRedirs Too many redirects</td>
</tr>
<tr>
<td>17</td>
<td>OffsiteRef Off-site or unapproved redirect or frame</td>
</tr>
<tr>
<td>18</td>
<td>UnkProtocol Unknown/unimplemented access method</td>
</tr>
<tr>
<td>19</td>
<td>BadParam Bad parameter</td>
</tr>
<tr>
<td>20</td>
<td>UnkErr Unknown error</td>
</tr>
<tr>
<td>21</td>
<td>BadRedir Bad redirect</td>
</tr>
<tr>
<td>22</td>
<td>DocUnauth Document access unauthorized</td>
</tr>
<tr>
<td>23</td>
<td>DocForbidden Document access forbidden</td>
</tr>
<tr>
<td>24</td>
<td>DocNotFound Document not found</td>
</tr>
<tr>
<td>25</td>
<td>ServerNotImplemented Server did not recognize request (unimplemented)</td>
</tr>
<tr>
<td>26</td>
<td>ServiceUnavailable Service unavailable</td>
</tr>
<tr>
<td>27</td>
<td>UnkMethod Unknown request method</td>
</tr>
<tr>
<td>28</td>
<td>CannotReadFile Cannot read from file</td>
</tr>
<tr>
<td>29</td>
<td>CannotLoadLib Cannot load dynamic library</td>
</tr>
<tr>
<td>30</td>
<td>ScriptErr Script error</td>
</tr>
<tr>
<td>31</td>
<td>ScriptTimeout Script timeout</td>
</tr>
<tr>
<td>32</td>
<td>ScriptMemExceeded Script memory limit exceeded</td>
</tr>
<tr>
<td>33</td>
<td>DisallowedProtocol Disallowed protocol</td>
</tr>
<tr>
<td>34</td>
<td>SslErr SSL error</td>
</tr>
<tr>
<td>35</td>
<td>ProxyUnauth Proxy access unauthorized</td>
</tr>
<tr>
<td>36</td>
<td>EmbeddedSecurityChange Embedded object security change</td>
</tr>
<tr>
<td>37</td>
<td>DisallowedFilePrefix Disallowed file prefix</td>
</tr>
<tr>
<td>38</td>
<td>DisallowedFileType Disallowed file type</td>
</tr>
<tr>
<td>39</td>
<td>DisallowedNonlocalFileUrl Disallowed non-local file URL</td>
</tr>
<tr>
<td>40</td>
<td>CannotConvertCharset Cannot convert character set</td>
</tr>
<tr>
<td>41</td>
<td>DisallowedAuthScheme Disallowed authentication scheme</td>
</tr>
<tr>
<td>42</td>
<td>SecureTransNotPossible Secure transaction not possible</td>
</tr>
<tr>
<td>43</td>
<td>UnexpectedResponseCode Unexpected server response</td>
</tr>
<tr>
<td>44</td>
<td>DisallowedMethod Disallowed request method</td>
</tr>
<tr>
<td>45</td>
<td>ConnUpgradeToSslRequired Connection upgrade to SSL required</td>
</tr>
<tr>
<td>46</td>
<td>FetchNotPermittedByLicense Fetch not permitted by license</td>
</tr>
<tr>
<td>47</td>
<td>UnknownContentEncoding Unknown Content- or Transfer-Encoding</td>
</tr>
<tr>
<td>48</td>
<td>DisallowedContentEncoding Disallowed Content- or Transfer-Encoding</td>
</tr>
<tr>
<td>49</td>
<td>CannotDecodeContentEncoding Cannot decode Content- or Transfer-Encoding</td>
</tr>
</tbody>
</table>
### DIAGNOSTICS

`urlinfo` returns the requested value(s).

#### EXAMPLE

```<fetch "http://www.somesite.com/mypage.html">
<urlinfo "metanames">
<$names = $ret>
Meta data:
<LOOP $names>
  <urlinfo "metaname" $names>
  $names = <LOOP $ret> "$ret" </LOOP>
</LOOP>
</fetch>```

#### CAVEATS

The `urlinfo` function was added in version 2.1.88480000 19980114.

If `submit` is used with `TOFILE`, then content and content-derived items such as `links` are unavailable in `urlinfo`, because the content was not held in memory for processing.

#### SEE ALSO

`fetch`, `submit`, `urlcp`
1.5. BUILTIN FUNCTIONS

1.5.39 urltext, urllinks – get formatted text, URL links

SYNOPSIS

$urltext$
$urllinks$

DESCRIPTION

The urltext function returns the formatted text of the last document fetched by fetch or submit. urllinks returns a list of the document’s non-image links, as fully-qualified (i.e. absolute) URLs.

Note: All information about the last fetched document – including formatted text and links – is available from the urlinfo function, p. 185, which is preferred over the obsolescent functions urltext and urllinks. See $urlinfo text$ (p. 193) and $urlinfo links$ (p. 189) for details.

DIAGNOSTICS

urltext returns the formatted text of the last-fetched document. urllinks returns a list of its non-image links, in absolute path form.

EXAMPLE

$<fetch "http://www.somesite.com/">
$urltext$
The text is:
$<PRE>
$ret$
</PRE>

CAVEATS

The urltext and urllinks functions were added Jan. 15 1997.

If submit is used with TOFILE, then urltext and urllinks will return nothing, because the content was not held in memory for processing.

SEE ALSO

fetch, submit, urlinfo, urlcp
### 1.5.40 urlcp – modify URL control parameters

#### SYNOPSIS

```
<urlcp $setting [$arg ...]>
```

#### DESCRIPTION

The `urlcp` function controls the behavior of the `fetch`, `submit` and `nslookup` functions. The `$setting` argument indicates the setting to change, which may take zero or more arguments given by `$arg` etc. Boolean settings take 1 boolean argument (e.g. “true”, “false”, “on”, “off”, “1”, “0”). Integer settings take 1 integer argument; in version 2.6.937800000 19990920 and later, an optional suffix “K”, “M” or “G”, for kilo, mega and giga, respectively, is permitted. String settings take 1 string argument. Only the first value of each argument is used, unless otherwise noted. The settings that are controlled by `urlcp` are as follows:

#### Resource Limits

The following `urlcp` settings control resources used by the fetch functions, such as time, size, memory and files:

- **maxframes (integer)**
  
  Sets the maximum number of subsidiary frames, iframes, and/or JavaScript pages `<SCRIPT SRC=...></SCRIPT>` to also fetch for a document. The default is 5. Frames are only fetched if `getframes` is on, `<IFRAME>`s only if `getiframes` is on, and JavaScript pages are only fetched if `javascript` and `getscripts` are on.

- **maxhdrsize (integer)**
  
  The maximum size of headers allowed; when exceeded a “Max headers size exceeded (truncated)” message is generated. The default is 128K; -1 indicates no limit. It is rarely necessary to increase this limit, as headers are small and mostly fixed in size. Added in version 2.6.937800000 19990920; was part of `maxpgsize` in prior versions. In version 4.01.1023500000 20020607 and later the previous setting is returned. Aka `maxhdrsz`, `maxheadersize`, `maxheadersz`.

- **maxconnidletime (double)**
  
  Maximum idle time in seconds (unused time between requests) that a connection may be open before it is closed and not reused for Keep-Alive. The default is 5. Added in version 5.00.1094074734 20040901. Returns previous setting.

- **maxconnlifetime (double)**
  
  Maximum seconds of real time that a connection can be open and still be re-used with Keep-Alive for future requests. Idle connections that are older than this will be closed. Defaults to 600 (i.e. 10 minutes). Added in version 5.00.1096313795 20040927. Returns previous setting.
1.5. BUILTIN FUNCTIONS

- **maxdownloadsize (integer)**
  Sets the maximum download size (network transfer size) of a response document body. -1 indicates no limit, and is the default. Maximum download size is checked before content and transfer encodings are decoded, i.e. it is a limit on network transfer size, not final document size (which the maxpgsize setting controls). A network transfer that exceeds maxdownloadsize will generate a “Max download size exceeded (truncated)” error message, and the transfer will be aborted. Added in version 5.01.1249039000 20090731. Aka maxdownloadsz.

  Note that since encodings are decoded on-the-fly as the document is downloaded, not only will exceeding maxdownloadsize abort the transfer, but typically so will exceeding maxpgsize—though possibly at an earlier point, if a compression encoding was used. This is why maxdownloadsize can usually be left at its -1 (unlimited) default, and just maxpgsize set instead; the latter setting controls final doc size, and indirectly sets an upper limit for network transfer bandwidth. Thus both memory and network usage can be limited with maxpgsize.

- **maxidleconn (integer)**
  Maximum number of idle connections to cache for future Keep-Alive requests. Added in version 5.1, where the default is 2. Returns previous setting.

- **maxkeepaliverequests or maxconnrequests (integer)**
  Sets the maximum number of page requests to do on a single Keep-Alive connection, before closing the connection and re-opening a new one. -1 sets no limit. Added in version 4.04.1068090000 20031105, where the default was 0, and Keep-Alive was only supported if <urlcp netmode sys> was set. In version 5.1, the default is 100, and Keep-Alive is supported for normal <urlcp netmode int> fetches as well. Setting 1 (or 0) turns off Keep-Alive, i.e. all requests send a Connection: close header and only one request per connection is used. Note that a value of 3 or more is needed for NTLM authentication (Integrated Windows Authentication) to function. Returns previous setting.

- **maxpgcachesz or maxpagecachesz or maxpgcachesize or maxpagecachesize or pagecachesz (integer)**
  Maximum page cache size in bytes. Only indirect pages are cached (e.g. frames, iframes, scripts), not directly <fetch>ed pages. Default is 5MB. Suffixes KB, MB, GB may be given, e.g. “5MB”. Returns previous setting. Added in version 5.00.1096313795 20040927.

- **maxpgsize (integer)**
  Sets the maximum page size that will be accepted for a document. Documents that are longer will be truncated and generate a “Max page size exceeded (truncated)” message. The default is 512KB; a setting of -1 indicates no limit. In versions prior to 2.6.93780000 19990920, the default was 100KB, the size of any headers was included, and -1 was not permitted. In version 4.01.1023500000 20020607 and later the previous setting is returned. Aka maxpagesize, maxpagesz, maxpgsz.

  The maximum page size is checked after all content and/or transfer encodings (if any) are decoded; i.e. it controls the size of the final document returned by <fetch>. Since encodings are decoded on-the-fly as the document is downloaded, reaching maxpgsize will also typically abort the network transfer as well. See also the maxdownloadsize setting for limiting network document size directly (though setting maxpgsize is typically enough).

- **maxprotspacecachesz or maxprotspacecachesize (integer)**
Maximum protection space cache size in bytes. The protection space cache is used to determine which URLs are protected with what user/pass credentials, so that later fetches to the same space do not need to negotiate credentials (and waste transactions). Defaults to 128KB. Suffixes KB, MB, GB may be given, e.g. “5MB”. Returns previous setting. Added in version 5.1.

- **maxprotspaceidletime (integer)**
  Maximum idle time of a protection space in the cache, in seconds. After this amount of time in disuse, the protection space will be deleted, which means future fetches may have to re-negotiate credentials. Defaults to 3600 (i.e. 1 hour); -1 is unlimited. Returns previous setting. Added in version 7.06.1465935000 20160614.

- **maxprotspacelifetime (integer)**
  Maximum lifetime of a protection space in the cache, in seconds. After this amount of time since its creation, the protection space will be deleted, which means future fetches may have to re-negotiate credentials. Defaults to -1 (unlimited). Returns previous setting. Added in version 5.1. Prior to version 7.06.1465935000 20160614, the default was 3600 (one hour).

- **maxredirs (integer)**
  Sets the maximum number of redirects that will be followed per URL fetch. Exceeding this limit generates the message “Too many redirections (N) while fetching ...”. The limit may be 0 to disallow redirects altogether. The default maxredirs value is 5.

- **pacfetchretrydelay (integer)**
  Sets the time in seconds before retrying a failed proxy pacurl fetch (p. 206), if enabled. Because all URLs fetched depend on the proxy auto-config script, all <fetch>es would keep attempting to re-fetch the same PAC URL if it fails. Thus, to reduce the load on the PAC URL server, the PAC script will not be re-fetched after error for pacfetchretrydelay seconds. The default is 10; a negative value means infinite (never retry automatically). Changing the pacurl will clear the last-try timestamp and allow a re-fetch to occur. Added in version 7.05.

- **proxyretrydelay (integer)**
  Sets the time in seconds before retrying a proxy when other proxies are available. When proxy auto-config is enabled (p. 206), it is possible that the FindProxyForURL() PAC function may return more than one proxy for a given URL: these proxies are normally tried in the order returned, until one succeeds. However, a “bad” (e.g. unresponsive) proxy is flagged in the proxy cache: every subsequent <fetch>'s FindProxyForURL()-returned list will be altered to have such bad proxies moved to the end of the list. This deprecation lasts proxyretrydelay seconds (or until the proxy succeeds).

  This allows successive <fetch>es to dynamically adapt to unresponsive proxies, even when the FindProxyForURL() list may be constant or unaware of the proxy’s unresponsiveness. For example, ifFindProxyForURL() always returns “PROXY flaky.example.com; PROXY reliable.example.com”, once the flaky.example.com proxy detectably fails, future fetches will try reliable.example.com first, for up to proxyretrydelay seconds, instead of waiting for flaky.example.com to fail first.

  However, ifFindProxyForURL() always returns just “PROXY flaky.example.com”, that proxy will always be tried, even after failure: there is no other proxy offered to try.
The `proxyretrydelay` setting was added in version 7.05, and defaults to 300. Negative values mean infinite, i.e. never automatically retry a bad proxy (when others are offered). The internal cache of bad proxies may be cleared with the `clearproxycache` option (p. 233).

- **savedownloaddoc (boolean)**
  Whether to save the network-transferred download doc, if it varies from the final (after content/transfer encodings decoded) document. The default is off to save memory, since the decoded document is usually more useful. Added in version 5.01.1249203000 20090802.

- **scriptmaxtimer (integer)**
  Sets maximum time (in seconds) to run JavaScript timers (set by `setInterval()` and `setTimeout()`). This represents a compromise between the dynamic JavaScript environment and the static return value of the fetch lib. The default is 3 seconds, which is less than the `scripttimeout` and allows some timers to run, but doesn’t wait indefinitely for an infinitely-recurring `setInterval()`. A value of -1 means no limit (but `scripttimeout` still applies). Returns previous value. Added in version 4.03.1050609000 20030417. See also `scriptrealtimers` (p. 225).

- **scriptmem (integer/size)**
  Controls how much memory (in bytes) to allow the JavaScript engine to allocate when running JavaScript code. Exceeding this limit may generate an error such as “JavaScript exceeded scriptmem limit”. This helps prevent erroneous JavaScript pages from consuming all available memory, e.g. if there is an infinite JavaScript loop. Standard memory size suffixes such as MB or KB may be appended to the integer value for clarity. The default value is 20MB. Note that a very low limit may cause problems even for pages with no JavaScript, as some JavaScript library objects must be allocated for every page; a minimum value of several MB is recommended. Returns previous setting. Added in version 4.01.1023500000 20020607.

- **scriptgcthreshold (integer/size/percentage)**
  Sets the threshold of `scriptmem` usage that the JavaScript engine should begin garbage collection. Can be a percentage (e.g. the default of “75%”), or an absolute integer/size (e.g. “15MB”). Added in version 7.06.1490209000 20170322.

- **scripttimeout (integer)**
  Controls how much total time (in seconds) to allow JavaScript code to execute on a page. Exceeding this limit will generate a “Timeout: JavaScript exceeded scripttimeout” message. This helps prevent an infinite loop in JavaScript from consuming all CPU and hanging the process. Note that this is a limit for the total time consumed by a page’s JavaScript, not per `<SCRIPT>` block. This timeout also applies after the page has been fetched, so it need not be smaller than the page `timeout`. The default is 5 seconds. -1 indicates no limit. Returns previous setting. Added in version 4.01.1023500000 20020607. Do not confuse this setting with the Vortex script timeout (p. 59), nor the `fetch` timeout (below).

- **timeout (integer)**
  Sets the per-fetch timeout, in seconds. A document fetch that takes longer than the timeout is aborted, the data read so far (if any) is returned, and an error message is issued (may be captured via `putmsg`, p. 581). The default is 30 seconds. This timeout applies to `nslookup`, and to each URL fetched by `fetch` and `submit`, so a framed document or one with `<SCRIPT SRC=...>` links, redirects etc.
may take longer. Do not confuse this setting with the Vortex script timeout (p. 59), nor the JavaScript timeout scripttimeout (above).

- **writebuffersize (integer)**
  Sets the initial buffer size to use for some writes, e.g. for writing <submit TOFILE> documents to disk, or decoding content/transfer encodings. The default is 32KB. Some write buffers may increase past this limit if needed. Added in version 5.01.1249039000 20090731. Returns previous buffer size.

**Page Fetching**

The following urlcp settings control how or whether pages and related URLs are fetched, such as frames and iframes:

- **encodings [add|del|set] [$encodings ...]**
  Sets the list of allowed content/transfer encodings for pages fetched. The $encodings argument(s) are zero or more of the values 7bit, 8bit, binary, identity, chunked, gzip, deflate or compress. The chunked encoding only applies to transfer encodings; the remainder apply to both content and transfer encodings. If the first value of the first argument is add, the given encoding(s) will be added to the allowed list; if del, deleted from it; if set, the list is cleared and set to $encodings (this is the default action if no add/del/set action is given). The keyword all may be used to refer to all encodings, and default may be used (with set) to re-set the default (which is identity, chunked, gzip, deflate and compress).

  The Vortex fetch library will declare the list of encodings it allows in Accept-Encoding and TE request headers, if httpversion is set to 1.1 (these are 1.1 headers, and some servers do not handle them as expected in a 1.0 request; httpversion is 1.1 by default in version 6 and later). It is up to the remote server to then choose encoding(s) from the declared list(s). The content encoding(s) (if any) of the returned document should be declared by the server in the Content-Encoding header, and transfer encoding(s) in the Transfer-Encoding header. Both types of encodings will be decoded before the document is returned from <fetch> or <urlinfo rawdoc>. If an encoding that is not allowed is encountered, a “Disallowed Content- or Transfer-Encoding” error is generated.

  Added in version 5.01.1249073000 20090731. Returns previous list of allowed encodings. See also the maxpgsize and maxdownloadsize settings for how they interact with encodings.

  7bit, 8bit and binary were added in version 7.03.1430243000 20150428. These are MIME Content-Transfer-Encoding values; some web servers (Apache) are known to use them as HTTP Content-Encoding values however.

- **fileexclude (list)**
  List of file trees to exclude (disallow) when fetching a local file:// URL. The default is none (no restrictions) for Windows, and “/dev/”, “/proc/” and “/debug/” for Unix. After fileroot is applied, if the resulting local file path from a file:// URL has one of these paths as a prefix, the URL will not be fetched. This can be used to protect certain unsafe or private directories on a local filesystem from being inadvertently walked. Does not apply to FTP-mapped non-localhost file:// URLs. Added in version 4.02.1048785087 20030327. Aka fileexcludes. Returns previous setting.
1.5. BUILTIN FUNCTIONS

- **fileinclude (list)**
  List of file trees to include (require) when fetching a local file:// URL. The default is none (no restrictions). After fileroot is applied, if the resulting local file path from a file:// URL does not have one of these paths as a prefix, the URL will not be fetched. This can be used to keep a local filesystem walk within certain directories. Does not apply to FTP-mapped non-localhost file:// URLs. Added in version 4.02.1048785087 20030327. Aka fileincludes. Returns previous setting.

- **filenonlocal (string)**
  How to handle non-localhost file:// URLs, i.e. ones with a specific host other than empty string or “localhost”. The value can be one of:
  - **off**
    Default: do not allow non-localhost file:// URLs. This ensures that no FTP or UNC paths are used.
  - **unc**
    Map non-localhost file:// URLs to their UNC paths and attempt to open as a local file. E.g. the URL “file://myhost/mydir/myfile” would map to the file “\myhost\mydir\myfile” under Windows and “//myhost/mydir/myfile” under Unix (but see modifications under fileroot below). This allows the behavior of web browsers that support UNC paths to be emulated on operating systems that support UNC, for consistency with browser views.
  - **ftp**
    Map non-localhost file:// URLs to FTP. E.g. the URL “file://myhost/mydir/myfile” would map to the URL “ftp://myhost/mydir/myfile” and be fetched as such. This allows the behavior of some browsers/operating systems that do not support UNC paths to be emulated.

  Added in version 4.02.1048785087 20030327. Returns previous setting. Note that filenonlocal only applies when a proxy is not set; when a proxy is active, all file:// URLs are passed to the proxy.

- **fileroot (string)**
  Sets the root directory to prepend to local file:// URL paths; default none. E.g. with fileroot set to “/docs”, the URL “file://localhost/dir/file.txt” would be read from the file “/docs/localhost/dir/file.txt”. Also applies to non-localhost URLs when filenonlocal is set to unc, e.g. the URL “file://myhost/mydir/myfile” is read from the file “/docs/myhost/mydir/myfile”. This allows both localhost and non-localhost file:// URLs to be mapped to a single directory hierarchy, perhaps where network filesystems corresponding to individual host(s) are mounted. Added in version 4.02.1048785087 20030327. Returns previous setting.

- **filetypes [add|del|set] [file|dir|device|symlink|other ...]**
  Sets the list of allowed file types for local file:// URLs. The possible values are file for ordinary files, dir for directories, device for devices, symlink for symbolic links (if supported by operating system), and other for other types (sockets etc.). If the first value of the first argument is add, the given list will be added to the allowed list; if del, deleted from; if set, cleared and set (the default). The default list is file, dir and symlink. If the file derived from a local file://
URL is not one of these types, it is disallowed. This prevents links to URLs like “file://localhost/dev/zero” from hampering a walk. Added in version 4.02.1048785087 20030327. Returns previous setting.

- `ftpactivepassivefallback (boolean)`
  If on (the default), FTP passive mode fetches will fall back to active mode on failure, and vice-versa. This may help resolve a fetch to an FTP server that does not support the current mode (or is firewalled), i.e. in cases where `ftppassive` is not set properly for the given situation. Only failures of the `PORT` or `PASV` command, or a temporary (5nn) error response to the main (`RETR/STOR/etc.`) command will trigger the mode switch. Added in version 6.00.1304040000 20110428. Returns previous setting.

  Note that if the correct mode (active or passive) is already known in advance, it is preferable to set it from the outset via the `ftppassive` setting, to avoid potential delays and/or errors from relying on this fallback switchover.

- `ftppassive (boolean)`
  If on (the default), FTP passive mode is used first for FTP protocol fetches. If off, FTP active mode is used first. Passive mode can be useful in situations where a firewall on the client (Vortex) side of the network prevents an FTP transfer (e.g. timeout). This is due to the nature of active-mode data transfers, where the remote (server) side is required to initiate a separate socket connection back to the client (even though the client initiates the original control connection). Many firewalls will block such incoming connections, causing the transfer to timeout. Passive mode allows the client to initiate both the control and data connections, which is often permitted by the client’s firewall. Added in version 5.01.1121350905 20050714. Note: Prior to version 6.00.1304040000 20110428 this setting was off by default. Returns previous setting.

  Note that if `ftpactivepassivefallback` is on (the default), the alternate mode may be used if the first mode (set by this setting) fails.

- `ftprelativepaths (boolean)`
  If on (the default), FTP paths are assumed to be login-dir-relative, so the URL “ftp://host/dir/file.txt” would be fetched with “RETR home/dir/file.txt” instead of “RETR /dir/file.txt” (where `home` is the FTP user’s login directory). For most (i.e. anonymous) FTP URLs this makes no difference, as the FTP login dir is typically at the root of the FTP-accessible tree. However, for many FTP URLs that require a true login, the FTP login dir is not the root dir, but the user’s home directory. Thus, with `ftprelativepaths` on, the above URL would fetch “dir/file.txt” from the user’s home directory – not “/dir/file.txt” from the root dir, where it may not exist. With `ftprelativepaths` off, the user’s home directory – which may be unknown or vary from user to user – would have to be specified in the FTP URL in order to get back to the FTP login dir.

  Dirs outside the FTP login dir may still be accessed when `ftprelativepaths` is on, however, by encoding an extra slash in the URL, e.g. “ftp://host/%2Fdir/file.txt”. Added in version 6. In previous versions the setting was effectively off.

- `ftpsendrelativepathsasabsolute (boolean)`
  If on (the default), relative FTP paths (i.e. due to `ftprelativepaths`) are changed to absolute paths when sent to the server, by prefixing the login directory (obtained with a `PWD` after login). This avoids the occasional need for a no-argument “CWD” command to go back to the login directory.
1.5. BUILTIN FUNCTIONS

(which some servers do not support), while still supporting the functionality of ftprelativepaths (no home dir needed in URLs). If off, the login directory is not prefixed; i.e. the URL “ftp://host/dir/file.txt” is fetched with “RETR dir/file.txt”. This setting has no effect if ftprelativepaths is off. Added in version 6.00.1301360000 20110328.

- getframes (boolean)
  If on, the <frame> objects of documents are fetched. The raw HTML returned will remain the same (the original document), but the formatted text from <urlinfo text> will be replaced and instead contain each frame in sequence. The links returned by <urlinfo links> or allrefs will be the list of all the frames’ links (e.g. the original URL – frame parent – will not have its links nor frames etc. included). The default is false, e.g. frames are not fetched. Note that only one level of frames is fetched, i.e. a <frame> link inside a <frame> link will not be fetched.

- getiframes (boolean)
  If on, inline <iframe> documents are fetched. The raw HTML returned will remain the same (the original document). The formatted text from <urlinfo text> will also remain, except that <iframe> blocks will be replaced with their referenced document text in-line. Note that like frames, only one level of <iframe>s is fetched. The <iframe> links are removed from the iframes, links, and allrefs lists returned by the urlinfo function. The default is false.

- getscripts (boolean)
  If on, and javascript is on, <SCRIPT SRC=...></SCRIPT> URLs on a page will also be fetched and run if they refer to JavaScript (and their URLs removed from the urlinfo links and allrefs lists). If off (default), such URLs are not fetched, and only inline <SCRIPT>...</SCRIPT> scripts are run (if javascript is on). Returns previous value. Added in version 4.01.1023800000 20020611.

- httpversion $version
  Sets the HTTP version to use for requests. The $version argument is one of 0.9, 1.0 or 1.1. HTTP/1.0 is the default for Texis/Webinator version 5 and earlier; HTTP/1.1 is the default for Texis/Webinator version 6 and later (and is only conditionally supported). It may be necessary to set 1.1 to fully utilize some features, e.g. content/transfer encodings (see the encodings setting, p. 202). Added in version 5.01.1249039000 20090731. Returns previous version.

- ignoreanchorframes (boolean)
  Whether to ignore frames and IFRAMEs that are just anchors, e.g. src="#". These usually just contain JavaScript, and fetching them just doubles up the content, links etc. of the parent URL. On by default. Added in version 6.

- inputfileroot (string)
  If set, all set/non-empty <input type="file"> values must be within this local directory tree (and not contain "../" components to get out of it), when <urlcp domvalue "...submitContent"> or variants are called. Value(s) that are outside this setting will cause an error such as 'Will not add form input ‘...’ file ‘...’ to submit content: Not in inputfileroot directory or contains ‘../’', and will be treated as empty (i.e. sent as empty value with no file). This is for security, to ensure all to-be-uploaded files are from a known directory. Added in version 6.00.1335222312 20120423. Default is unset (i.e. no check is performed). Returns 1 on success, 0 on error.
• linkprotocols [add|del|set] [$protocols|allowed ...]
  Sets the list of protocols allowed to be returned in links from a page (i.e. the links value of the
  urlinfo function, p. 189). Note that this setting does not control what can be fetched, only the list
  of links returned from a page. It can be used as a filter to remove invalid-protocol links returned by a
  page. The $protocols argument(s) are a list of zero or more values, each of which is either a
  recognized protocol (see protocols below), the value unknown for unknown protocols, or the
  value allowed for just protocols permitted by the protocols setting. The default is all protocols
  plus unknown. If the first value of the first argument is add, the given list will be added to the
  allowed list; if del, deleted from; if set, cleared and set (the default). Returns previous setting.
  Added in version 4.01.1029180431 20020812.

• methods [add|del|set] [$methods ...]
  Sets the list of request methods allowed for page fetching (default all). The $methods argument(s)
  are zero or more of the values OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, MKDIR,
  RENAME, SCHEDULE, COMPILE or RUN. Not all methods are supported by all protocols; e.g.
  MKDIR is only supported by FTP. If the first value of the first argument is add, the given method list
  will be added to the allowed list; if del, deleted from; if set, cleared and set (the default).
  Alternately, the default methods may be restored with set default. Returns previous setting.
  Added in version 5.01.1232696000 20090123.

• netmode (string)
  Sets the routines to use for page fetching. The default is int., which uses Texis’ internal routines. For
  Windows versions, netmode may be set to sys, which uses the system routines. This may allow
  certain authenticated sites to be accessed, if the internal routines’ NTLM authentication is not
  sufficient for example. However, parallelization and many other settings and features are not
  unavailable. Added in version 4.04.1068000000 20031104.

• offsiteok (boolean)
  If on (default), documents that are off-site from the original URL (e.g. redirects) will be fetched if
  needed. If false, such redirects will not be fetched.

• protocols [add|del|set] [$protocols ...]
  Sets the list of URL protocols allowed to be fetched. $protocols is a list of zero or more of the
  supported protocols http, ftp, gopher, javascript, https or file. The default list is
  http, ftp, gopher, javascript and https. (Note that javascript must be also on if
  JavaScript URLs are to work.) If the first value of the first argument is add, the given list will be
  added to the allowed list; if del, deleted from; if set, cleared and set (the default). Returns the
  previous list of allowed protocols. Added in version 4.01.1024300000 20020617. file support
  added in version 4.02.1048785087 20030327. Note that changing these protocols may affect what
  links are returned by <urlinfo links> (p. 189), if linkprotocols (p. 205) is allowed.

• proxy [type] $proxyUrl
  Takes a URL as argument. The protocol, host and port of $proxyUrl will be used as the proxy or
  tunnel to fetch all future URL requests (except for javascript: URLs, which are always evaluated
  internally as they are source code, not resource locations). The $proxyUrl protocol must be HTTP
  or (in version 4.02.1048785087 20030327 and later) HTTPS. In version 4.04.1077500000 20040222
  and later, an empty string value will clear the proxy, i.e. turn it off and resume direct connections.
  In version 7.05 and later, a type may be given; one of:
1.5. BUILTIN FUNCTIONS

– pacurl
The URL given is a proxy auto-config URL, i.e. a JavaScript proxy.pac file containing a
FindProxyForURL(url, host) function that will return the proxy(ies) to use for a given
URL. The URL MIME type should be application/x-ns-proxy-autoconfig. The
PAC script will be automatically fetched at the next <fetch> or <submit> statement, as
needed.

– pacscript
The URL argument is instead the PAC script itself.

If proxy auto-config is enabled via either of these types, the script’s FindProxyForURL() function will be called for every URL. This function returns a list of one or more proxies to use for the
given URL (or DIRECT to not use a proxy). Thus, proxy auto-config allows URL-by-URL
customization of proxies, and/or organization-wide proxy configuration. The PAC script is run in a
restricted environment; see findproxyforurl.com for details on the JavaScript functions
available to PAC scripts.

Failure to fetch the pacurl script will cause the current <fetch> to fail with PacError, as well
as all subsequent <fetch>es until pacfetchretrydelay expires (p. 200). Proxies that are
deemed “bad” (e.g. unresponsive) will have a lower priority for proxyretrydelay seconds
(p. 200).

An optional proxy mode argument (same as proxymode) may also be given after the URL, if no
type is specified. This syntax is for back-compatibility and is deprecated in favor of the
proxymode setting; it may be removed in future releases.

• proxymode $mode
Determines how to use a proxy (if set). $mode is one of:

  – auto
    Automatically select proxy or tunnel mode depending on the requested URL: tunnel for HTTPS,
    otherwise proxy. This is the default if $mode is not specified.

  – proxy
    Always use proxy mode, i.e. tell proxy to GET (or whatever method was requested) the
    requested absolute URL.

  – tunnel
    Always use tunnel mode, i.e. tell proxy to CONNECT to the requested host and port, then
    proceed with request as if directly connected to the requested server. If the request URL’s
    protocol cannot be tunneled (e.g. file: or FTP), the request fails.

Added in version 7.05. In versions prior to 7.00.1363052000 20130311, the mode was always
proxy, even for javascript: URLs. In version 7.00.1363052000 20130311 through 7.04, the
default mode was auto.

Note that an HTTPS tunnel to an HTTPS origin server is not currently supported by the fetch library
(an HTTPS proxy to an HTTPS server is supported, however). Many tunnels do not support an
HTTPS tunnel to an HTTP origin server, and some proxies do not support HTTPS origin servers
(since the proxy would then have to provide a certificate etc.).
Server Authentication

The following `urlcp` settings control authentication information sent to the web server (e.g. for restricted-access pages):

- `authenticateservermutually` (boolean)
  Whether to attempt to also mutually authenticate the server during a 401 Unauthorized authentication transaction. Experimental; may not be supported; may change or be dropped in future releases. Only supported in Negotiate authentication. Added in version 7.04. Default off.

- `authschemes [add|del|set] [$schemes ...]`
  Sets which authentication schemes to use. $schemes can be one or more of `anonymous` (i.e. none), `FTP`, `Basic`, `file`, `Negotiate`, `NTLM`, `NTLMv1` or `NTLMv2`. (Authentication via SSL client certificates is handled at the SSL, not HTTP level; see the `sslcertificatefile` setting, p. 213.) NTLM is an alias for both NTLMv1 and NTLMv2 together. Returns previous list. The default is all of the above, except for NTLMv1 (and Negotiate, if not supported on current platform). If the first value of the first argument is `add`, the given list will be added to the allowed list; if `del`, deleted from; if `set`, cleared and set (the default). Added in version 5.1. NTLMv2 support was added in version 5.01.1222370000 20080925. Note that only flag negotiation is currently supported, as full NTLM session security is not used in NTLM over HTTP. The default is `ignore`.

If both NTLMv1 and NTLMv2 are enabled, NTLMv2 will be attempted first, then NTLMv1 if that fails. NTLMv2 is more secure than NTLMv1, and is supported by servers running Windows NT4 SP4 or later OSes. **Note:** Windows servers or domain controllers which have the Local Security policy Network security: LAN Manager authentication level set to “Send NTLMv2 response only/refuse LM & NTLM” (or the registry setting `lmCompatibilityLevel` set to 5) will only accept NTLMv2, and will thus require NTLMv2 to be enabled. With NTLMv2 disabled via `<urlcp>`, such servers will reject a `<fetch>` (even with proper credentials) with 401 Unauthorized or 500 Internal Server Error.

Similarly, certain Samba implementations or configurations (as servers or domain controllers) may only understand NTLMv1, and may return 401 Unauthorized unless NTLMv1 is specifically enabled via this setting.

- `ntlm128bitencryption` (string or integer)
  Sets whether to negotiate 128-bit encryption during NTLM session security, for NTLMv1 and NTLMv2 authentication. The value is one of:
  - `ignore` Do not request or do 128-bit encryption
  - `accept` Do not request 128-bit encryption, but accept if peer offers
  - `request` Request 128-bit encryption, but optionally
  - `require` Request 128-bit encryption, and fail with error if peer refuses to do 128-bit

Returns previous setting, as an integer 0-15 (bits 0-1 apply to NTLMv1, bits 2-3 apply to NTLMv2). Added in version 5.01.1222370000 20080925. Note that only flag negotiation is currently supported, as full NTLM session security is not used in NTLM over HTTP. The default is `ignore`. 
1.5. BUILTIN FUNCTIONS

- `ntlmntlmv2sessionsecurity (string or integer)`
  Sets whether to negotiate NTLMv2 session security\(^{31}\) during NTLMv1 and NTLMv2 authentication. Same values accepted and returned as for `ntlm128bitencryption`. Added in version 5.01.1222370000 20080925.  
  Enabling NTLMv2 session security can prevent a 500 Internal Server Error response when accessing Windows servers whose Local Security policy for Network security: Minimum session security for NTLM SSP based (including secure RPC) servers is set to include “Require NTLMv2 session security”. Thus, the default for this setting is request.

- `ntlmsealing (string or integer)`
  Sets whether to negotiate NTLM sealing (message confidentiality) during NTLMv1 and NTLMv2 authentication. Same values accepted and returned as for `ntlm128bitencryption`. Added in version 5.01.1222370000 20080925. Note that only flag negotiation is currently supported, as NTLM sealing is not used in NTLM over HTTP. The default is `ignore`.

- `ntlmsigning (string or integer)`
  Sets whether to negotiate NTLM signing (message integrity) during NTLMv1 and NTLMv2 authentication. Same values accepted and returned as for `ntlm128bitencryption`. Added in version 5.01.1222370000 20080925. Note that only flag negotiation is currently supported, as NTLM signing is not used in NTLM over HTTP. The default is `ignore`.

- `ntlmv1128bitencryption (string or integer)`
- `ntlmv1ntlmv2sessionsecurity (string or integer)`
- `ntlmv1sealing (string or integer)`
- `ntlmv1signing (string or integer)`
  Same as `ntlm...` settings, but only affecting NTLMv1 transactions, not NTLMv2. Return value is previous setting, as integer 0-7. Added in version 5.01.1222370000 20080925.

- `ntlmv2128bitencryption (string or integer)`
- `ntlmv2ntlmv2sessionsecurity (string or integer)`
- `ntlmv2sealing (string or integer)`
- `ntlmv2signing (string or integer)`
  Same as `ntlm...` settings, but only affecting NTLMv2 transactions, not NTLMv1. Return value is previous setting, as integer 0-7. Added in version 5.01.1222370000 20080925.

- `pass (string)`
  Sets the password to use when accessing documents. The default is “user@host” for FTP, none for other protocols.

- `proxypass (string)`
  Sets the proxy password to use when accessing documents via a proxy. The default is none. Added in version 4.01.1031169922 20020904.

---

\(^{31}\)“NTLMv2 session security” is a somewhat confusing misnomer, since it can be used under NTLMv1 as well as NTLMv2. The term refers to extended session security (for later in the transaction) that has limited or no applicability to NTLM authentication (at transaction start). The session aspects of NTLMv2 session security appear not to be used under HTTP; only the authentication-time aspects, i.e. flag negotiation and changes to NTLMv1 authentication responses. In fact, for NTLMv2 transactions (over HTTP), the Windows Local Security policy Require NTLMv2 session security has no effect other than requiring that the flag be present.
• **proxyuser (string)**
  Sets the proxy user to use when accessing documents via a proxy (the `Proxy-Authorization: Basic` header). The default is none. Added in version 4.01.1031169922 20020904.

• **saslmechanisms [add|del|set] [$mechanisms ...]**
  Set or alter the list of SASL mechanisms to enable. The SASL API is used for Negotiate HTTP authentication on Unix platforms (where supported). The default list is GSSAPI, which supports Kerberos. The possible mechanism values depend on the SASL implementation, and typically include GSSAPI, NTLM and others; call `<urlinfo saslmechanismsavailable>` (p. 191) to obtain the list. (Note that NTLM here refers to a SASL mechanism under `WWW-Authenticate: Negotiate HTTP authentication`, not `WWW-Authenticate: NTLM HTTP authentication`.) If the first value of the first argument is `add`, the given list will be added to the allowed list; if `del`, deleted from; if `set`, cleared and set (the default). Added in version 7.04. Returns 1 on success, 0 on error (e.g. SASL not supported on platform). Note that invalid mechanisms may not be detected until a fetch using the SASL API is run. A `putmsg` is generated if SASL is not supported on the current platform.

• **saslpluginpath (string)**
  Sets the colon-separated path of directories to look for SASL plugins. The SASL API is used for Negotiate HTTP authentication on Unix platforms (where supported). The default is the `etc/sasl/lib/sasl2` subdir of the Texis install dir. Added in version 7.04. Returns 1 on success, 0 on error (e.g. SASL not supported on platform). Note that an invalid path may not be detected until a fetch using the SASL API is run. A `putmsg` is generated if SASL is not supported on the current platform.

• **sendlmresponse (boolean)**
  Whether to send the LM response in NTLM authentication responses. The LM response is used for backwards-compatibility with older Windows servers and domain controllers, but is considered less secure if the NTLMv1 protocol is used (NTLMv2 is used by default). On by default. Added in version 5.01.1213740000 20080617.

• **sendntlmresponse (boolean)**
  Whether to send the NTLM response in NTLM authentication responses. On by default. Added in version 5.01.1213740000 20080617.

• **sspipackages [add|del|set] [$packages ...]**
  Set or alter the list of SSPI packages to enable. The SSPI API is used for Negotiate HTTP authentication on Windows platforms. The default list is empty, which supports Kerberos and NTLM. The possible package values depend on the SSPI implementation; call `<urlinfo ssspipackagesavailable>` (p. 193) to obtain the list. (Note that NTLM here refers to an SSPI package under `WWW-Authenticate: Negotiate HTTP authentication`, not `WWW-Authenticate: NTLM HTTP authentication`.) If the first value of the first argument is `add`, the given list will be added to the allowed list; if `del`, deleted from; if `set`, cleared and set (the default). Added in version 7.04. Returns 1 on success, 0 on error (e.g. SSPI not supported on platform). Note that invalid packages may not be detected until a fetch using the SSPI API is run. A `putmsg` is generated if SSPI is not supported on the current platform (e.g. non-Windows).
Note that Negotiate authentication using SSPI is not fully implemented yet; this (and other SSPI) functions may not yet work correctly or may be updated/changed in future releases.

- **user (string)**
  Sets the user to use when accessing documents (e.g. Basic or NTLM authentication for HTTP pages, login user for FTP documents, LogonUser() for file:// documents). The default is “anonymous” for FTP, none for other protocols.

Windows file:// user/pass support was added in version 5.01.1123050000 20050803. For file:// URLs, the Windows operating system requires that the user specified must have Log on as a batch job permission, and the source user (the user running the Vortex script and calling <fetch>, typically I_USR if from a web server) must have Act as part of the operating system permission. **Note: before changing these permissions, discuss the security implications with your system administrator.** Lack of the former permission may result in Win32 error 1385 (ERROR_LOGON_TYPE_NOT_GRANTED: “Logon failure: the user has not been granted the requested logon type at this computer”), whereas lack of the latter permission may result in Win32 error 1314 (ERROR_PRIVILEGE_NOT_HELD: “A required privilege is not held by the client”).

### SSL/HTTPS

The following urlcp settings control SSL/HTTPS-related behavior of <fetch>/<submit>.

- **addentropy $entropy [$fraction]**
  Uses random data in string $entropy to help seed random number generator for SSL/HTTPS plugin initialization. $fraction is the fraction of $random considered to be truly random; it is a floating-point value from 0.0 to 1.0 (default 1.0). Only some Unix platforms need entropy for SSL startup, and it is usually obtained from the prngd daemon; this setting is a last-ditch fallback and should not be needed in a production environment. Note that providing non-random data to addentropy can decrease the security of SSL/HTTPS connections. Added in version 4.01.1028146532 20020731.

- **embedsecurity (string)**
  Sets security level for embedded URLs (e.g. frames, scripts) in a page. This controls whether or not to fetch such embedded objects based on the relative security (HTTPS or not) of the main URL and the embedded object URL. Possible values are:

  - **off**
    Default: fetch any embedded object URL as requested.

  - **nodecrease**
    Do not fetch non-HTTPS embedded objects if the main page is HTTPS.

  - **noincrease**
    Do not fetch HTTPS embedded objects if the main page is non-HTTPS.

  - **sameprotocol**
    Do not fetch embedded objects unless they are the same protocol (http, https, ftp etc.) as the main page.
Note that currently, this setting only treats HTTPS URLs as secure; it does not check if HTTP URLs were upgraded to SSL via the Upgrade header. Returns previous setting. Added in version 4.01.1031348302 20020906. See also secure setting, p 212.

- **entropypipe (string)**
  Sets the Unix file pipe path used to access the entropy daemon, when initializing the SSL/HTTPS plugin on some platforms. (The entropy daemon is only used when needed on Unix platforms that do not have /dev/random.) The default is the [Texis] Entropy Pipe setting in texis.ini (p. 578), or if that is not set, etc/egd-pool in the install directory. The value none indicates that no pipe is set, i.e. the entropy daemon should not be accessed. Returns 1 on success, 0 if failure. Added in version 4.01.1031605926 20020909. See also the prngdpid value of the urlinfo function (p. 190).

- **secure (string)**
  Controls whether pages are fetched securely (via SSL) or not, and whether to attempt to upgrade security if needed. Possible values are:

  - **off**
    Security is not needed. Pages will be fetched using the security provided by the URL/proxy protocol, if any; i.e. only HTTPS fetches will be secure. This is the default.

  - **preferred**
    Security is preferred but not required. If a non-secure protocol is used and supports upgraded security, the upgrade attempt will be made, but if the upgrade fails, or only part of the transaction can be made secure (e.g. response but not request), the rest of the fetch will still be made normally (insecurely). Currently only HTTP fetches can be upgraded to SSL, via the RFC 2817 Upgrade method.

  - **required**
    Security is required. Insecure protocols will only be permitted if they can be fully upgraded to secure (before the main URL request is made), otherwise the fetch will fail. Only HTTPS fetches, and HTTP with a successful Upgrade, are possible.

Note that for this setting, security applies to the over-the-wire protocol; i.e. an FTP URL can still be fetched securely using an HTTPS proxy. Security also applies to the first-hop connection, i.e. Vortex <fetch>: a proxy may or may not have a secure connection from itself to the requested resource.

When upgrading an HTTP connection to SSL via the OPTIONS-Upgrade RFC 2817 method, the OPTIONS method is used as a “no-op” request first, to negotiate the connection secure before the main URL request is sent. Upgrading to SSL requires Upgrade header, OPTIONS method, and Keep-Alive connection support by both the client (Vortex) and the server: if any of these are disabled or unsupported, the transaction may be partially or wholly insecure. For example, if the OPTIONS method is disabled via <urlcp methods> (p. 206), the SSL upgrade would have to be attempted on the main request, instead of the no-op OPTIONS request: if secure is required, this is not allowed and the fetch fails. But if secure is preferred, this is permitted, and although the request will be insecure, at least the server response is secure (if the upgrade succeeds). This can be a way of getting at least a partially-secure transaction from a server that does not support OPTIONS.

If a transaction cannot be conducted wholly securely and secure is set to required, an error such as Will not fetch URL http://...: Secure transaction not possible:
... will occur. Reasons include:

- TLS/SSL Upgrade was ignored, failed or insecure
  An Upgrade to SSL failed.

- TLS/SSL Upgrade via OPTIONS not possible: OPTIONS disallowed
  The OPTIONS method is disallowed (via <urlcp methods>, p. 206), and is needed to make
  the main request secure (by upgrading before the main request).

- Main request will not be possible on same connection if TLS/SSL
  Upgraded via OPTIONS: maxconnrequests <= 1
  The connection would have to be closed after the OPTIONS upgrade – before the main request
  can be made securely – since <urlcp maxconnrequests> is less than or equal to 1.

The secure setting returns its previous value. See also the secure option to <urlinfo>

- **sslcertificatefile $certFile**
  Sets SSL client (Vortex) certificate to PEM-encoded file $certFile; returns 1 on success, 0 on
  error. The certificate will be sent to HTTPS/SSL servers that specifically request it upon contact. (Not
  all servers do; client certificates are usually only requested if the server has SSL client authentication
  enabled.) An empty or unset $certFile value will clear the existing certificate (if any). Any
  pre-existing (cached/Keep-Alive) SSL connections will be terminated, so that the new certificate will
  be used.

  Note that if a certificate is set, the corresponding private key must also be set with the
  sslcertificatekey setting, or errors such as “SSL certificate set without
  certificate key: secure connections may fail” and “Cannot complete
  SSL handshake with host: ... alert handshake failure” may result during
  fetches. (The same PEM file may be used to contain both the certificate and key; however the
  sslcertificatekeyfile setting must still also be called.)

  If the server being connected to does authentication of client certificates, and the $certFile
  certificate is rejected, an error such as “Cannot complete SSL handshake with host:
  ... alert bad certificate”, “... alert certificate unknown” or “alert
  unknown ca” may result.

  The sslcertificate setting was added in version 6.00.1320460000 20111104. The default is
  unset (no client certificate). Note that setting a client certificate is not generally needed; it is usually
  only used when contacting an HTTPS server that does SSL client authentication.

- **sslcertificate $certString**
  Same as sslcertificatefile, but loads from a PEM string $certString instead of a file.

- **sslcertificatekeyfile $keyFile [$password]**
  Sets SSL client (Vortex) certificate private key to PEM-encoded file $keyFile: this key must
  correspond to the client certificate loaded with the sslcertificatefile setting. Returns 1 on
  success, 0 on error. An empty or unset $keyFile will clear the existing key (if any). The optional
  $password argument gives the password used to decrypt the key, if encrypted. Any pre-existing
  (cached/Keep-Alive) SSL connections will be terminated, so that the new key will be used.

  Note that if a key is set, a certificate must also be set with the sslcertificatefile setting, or
  errors such as “SSL certificate key set without certificate: secure
connections may fail” and “Cannot complete SSL handshake with host: ...
alert handshake failure” may result during fetches. (The same PEM file may be
used to contain both the certificate and key; however the sslcertificatefile setting must still
also be called.)

If the key is encrypted and the $password argument is incorrect, an error such as “Cannot
parse SSL certificate key: Bad password” may result. If the key is encrypted and
the $password argument is missing, an error such as “Cannot obtain password to
decrypt SSL certificate key: Missing or empty <urlcp
sslcertificatekey> password argument” may result. If the key does not match the
certificate set with sslcertificate, an error such as “Cannot use SSL certificate
key: ... key values mismatch” may result during fetches.

The sslcertificatekey setting was added in version 6.00.1320460000 20111104. The default
is unset (no client certificate private key). Note that setting a client certificate and key is not generally
needed; it is usually only used when contacting an HTTPS server that does SSL client authentication.

• sslcertificatekey $keyString [$password]
  Same as sslcertificatekeyfile, but loads key from PEM string $keyString instead of a
  file.

• sslcertificatechainfile $chainFile [skipfirst]
  Sets the chain for the sslcertificatefile client certificate to the PEM-encoded CA
certificate(s) in file $chainFile, in order from issuer to root. Returns the number of certificates
loaded into the chain (i.e. 0 or more), or -1 on error (e.g. file not found). (Note that it is not an error
for there to be no certificates in the chain file.)

  A certificate’s chain is the list of zero or more CA (certificate authority) certificates that form its
“pedigree”, i.e. so its authenticity can be verified and it can be trusted by peers – in this case by
servers, since this chain is for the client (Vortex) certificate. A chain starts with the certificate of the
CA that issued (signed) the leaf certificate (sslcertificate in this case), followed by the
certificate of the CA that issued that CA certificate, etc. up to a root (self-signed) CA certificate. Any
certificate that was issued by another certificate (i.e. is not self-signed) needs a chain.

  HTTPS/SSL servers that do client authentication may need the client’s chain to be able to follow it up
to a root certificate that they trust: otherwise they cannot verify the client certificate and may
terminate the connection with an SSL error (that may mention “alert bad certificate”,
“alert certificate unknown”, or “alert unknown ca”).

  Note that if no chain is set with sslcertificatechainfile – but the
sslcertificatefile certificate needs one – the Vortex fetch library will attempt to
automatically complete the chain with certificates from the sslcacertificatefile list, if
possible. Since sslcacertificatefile certificates are trusted but
sslcertificatechainfile certificates are not, if the client chain certificates are not to be
trusted (i.e. for verification of servers) they should be set with sslcertificatechainfile and
not sslcacertificatefile.

If the skipfirst option is given, the first certificate in $chainFile is skipped. This facilitates
using an “all-in-one” PEM file that contains both the client certificate and its chain certificate(s): the
same file can be given to both sslcertificatefile and sslcertificatechainfile,
with the skipfirst option for the latter.
The `sslcertificatechainfile` setting was added in version 6.00.1320460000 20111104. The default is unset (chain auto-completed from `sslcacertificates` or no chain). Note that a chain is only needed if a certificate is set.

- `sslcertificatechain $chainString [skipfirst]`
  Same as `sslcertificatechainfile`, except that the chain is loaded from the PEM string `$chainString` instead of a file.

- `sslcacertificates $certString`
  Sets the list of trusted CA certificates to the PEM-encoded certificates in file `$certString`. Returns the number of certificates read (0 or more), or -1 on error. (Note that it is not an error for there to be no certificates in the file.)

  Trusted certificates are used when verifying peer certificates (server certificates in this case): the server certificate chain’s root certificate must be trusted (i.e. included in this setting) for the server certificate to be verified by the Vortex fetch library. See `sslverifyserver` (p. 215) for more details on verification of server certificates.

  Trusted certificates are also used to automatically complete certificate chains if needed (both server and client). See `sslcertificatechainfile` (p. 214) for why it may not be best practice to implicitly set `client` (Vortex) chain certificates via `sslcacertificates`.

  The `sslcacertificates` setting was added in version 6.00.1320460000 20111104. The default is unset (no trusted certificates). Note that trusted certificates are only needed if `sslverifyserver` is not off.

- `sslcacertificate $certString`
  Same as `sslcacertificates`, except that the trusted certificate(s) are loaded from the PEM string `$certString` instead of a file.

- `sslverifyserver $onOffOrFlags`
  Enables or disables verification of HTTPS/SSL server certificates during fetches. Verification is enabled if `$onOffOrFlags` is `on`, or disabled if `off` (the default, which can also be set with an empty or unset value). Returns 1 on success, 0 on error (i.e. bad setting value). This verification is similar to what browsers do when connecting to a secure server.

  If verification of server certificates is enabled, and a server certificate cannot be successfully verified upon connecting to the server, the connection will be terminated with the error “Cannot verify certificate from host:port: reason at depth N”, and `<urlinfo errtoken>` will return `CannotVerifyServerCertificate`. The `reason` may vary (e.g. certificate has expired, unable to get issuer certificate, etc.); the full list is in the SSL Client/Server Certificate Verification appendix (p. 605). The depth `N` given in the message is the chain depth, or number of chain certificates away from the server certificate that the error happened. Thus depth 0 is the server certificate itself; depth 1 is the server certificate’s issuer certificate, etc.

  In addition to `on` (perform all verification checks) or `off`, the `sslverifyserver` setting may be set to the Apache-compatible values `require` (same as `on`), `none` (same as `off`), or `optional` (same as `on -No_Peer_Certificate`). Also, any number of individual verification errors may be enabled/disabled, by listing them, space-separated, after the initial `on/off/etc.` value, each with a plus or minus sign for enable or disable; see p. 605 for details. Note that disabling individual `sslverifyserver` errors like this should be done with caution, as it can weaken the security provided by the full-check `on` level.
The `sslverifyserver` setting was added in version 6.00.1320460000 20111104. It defaults to off. See the SSL Client/Server Certificate Verification appendix (p. 603) for more on certificate verification.

- **sslverifydepth $depth**
  Sets the maximum chain depth to allow when verifying an HTTPS/SSL server certificate. This is the maximum number of CA certificates (beyond the server certificate itself) to allow in the chain. I.e. a depth of 0 only allows a self-signed server certificate, a depth of 1 only allows a single CA chain certificate, etc.; a depth of -1 indicates no limit. Exceeding the verify depth results in a “Cannot verify certificate from host:port: certificate chain too long at depth $N” error message and the connection fails.

  The `sslverifydepth` setting was added in version 6.00.1320460000 20111104. It defaults to 1; however chain depth is only verified if `sslverifyserver` is on and the latter defaults to off. Note that a depth of -1, disabling the X509_V_ERR_CERT_CHAIN_TOO_LONG error in `sslverifyserver`, or turning off `sslverifyserver` completely, will disable server chain-depth checks.

- **sslciphers [$group] $cipherList**
  Which SSL ciphers to allow. The `$cipherList` argument is a string in OpenSSL cipher list format detailing the SSL ciphers to permit. It is a list of one or more cipher strings, separated by colons. Ciphers may be removed with “-“ (and never added again with “!“), or added with “+“. See https://www.openssl.org/docs/apps/ciphers.html#CIPHER-LIST-FORMAT for details on cipher strings and syntax. Returns nonzero on success, 0 on error.

  Changing the SSL cipher list may be used to remove weak or compromised ciphers, or to reduce the size of the SSL ClientHello message to avoid timeouts when connecting to long-handshake-intolerant servers. Added in version 7.03.1435875000 20150702. Note that some ciphers (e.g. considered weak or vulnerable) may be unavailable in later versions of Texis, depending on the OpenSSL version in use by that Texis release. Note also that setting an invalid cipher list may not result in an error until an HTTPS connection is actually attempted.

  In version 7.07 and later, an optional cipher `$group` may be given, to set the cipher list for that protocol group. The group may be SSL (the default) for protocols TLSv1.2 and below, or TLSv1.3 for TLSv1.3 ciphers; the two lists are independent.

- **sslprotocols [add|del|set] [\$sslprotocols ...]**
  Which SSL protocol(s) to allow. The `$sslprotocols` argument(s) are zero or more of the values SSLv2, SSLv3, TLSv1, TLSv1.1, TLSv1.2, TLSv1.3 or all for all protocols. If the first value of the first argument is add, the given protocol list will be added to the allowed list; if del, deleted from; if set, cleared and set (the default). Alternately, the default SSL protocols may be restored with set default. Returns previous setting. Added in version 5.01.1184873000 20070719. The default is TLSv1, TLSv1.1, TLSv1.2. (Prior to version 7.02.1413403000 20141015, the default was SSLv3 and TLSv1. Prior to version 7.3, TLSv1.1 and TLSv1.2 were unsupported. Prior to version 7.07 TLSv1.3 was unsupported.)

- **sslusesni $onOff**
  Whether to use SNI (Server Name Indication) when the TLS protocol (successor to SSL) is used. When on (the default), the server hostname is set at the TLS level, enabling the remote server to send the right certificate if it has multiple certificates for different hostnames. When off, the hostname is
not set, and connecting to some HTTPS servers that require SNI may fail with the error ...
alert handshake failure. Returns previous setting. Added in version 7.03.1426024000
20150310. In previous versions SNI was effectively off.

Formatted Text

The following urlcp settings control how HTML documents are formatted (e.g. the return value of
$urlinfo text$ etc.):

- **8bithtml** (boolean)
  If true (default), 8-bit HTML characters are left alone when formatting HTML text. If false, 8-bit
  characters are replaced with the closest 7-bit character(s).

- **allowinputfiledefault** (boolean)
  If true, $<input type="file">$ default values (i.e. those assigned in the original HTML, as
  opposed to those set by $<urlcp domvalue>$) will be allowed. If false (the default), such default
  values will be suppressed (i.e. empty, as if unset). This is for security, to help prevent malicious
  HTML from surreptitiously “stealing” local files by pre-setting file-upload dialogs. Added in version
  6.00.1335222312 20120423. Returns previous value (1 or 0).

- **allowpunct** (boolean)
  Sets whether to allow punctuation in tag/attribute names when parsing HTML. Added in version
  4.0.1001550000 20010926. Default is on, which aids in parsing of XML-like attributes.

- **alttxt** (boolean)
  If true (default), the text from ALT attributes in IMG and AREA tags is included in the formatted text.
  If false, this text is ignored. This is useful when the ALT text is “gif” or “image” or something equally
  inane.

- **charsetconfigfromfile** (string)
  Load charset configuration from the given file. The file format is a set of charset names, each
  followed by zero or more space-separated aliases:

  Charset: ISO-8859-1
  Aliases: 8859-1 CP819 csISOLatin1 IBM819
  Aliases: ISO_8859-1:1987 iso-ir-100 11 latin1
  ...

 Charsets encountered during fetch processing that match names in Aliases are canonicalized to
  their official Charset name. The default charset config file is set in texis.ini by the
  [Texis] Charset Config setting (p. 578). If that is unspecified, the file
  conf/charsets.conf in the install dir is used. Returns 2 on success, 1 on partial success (file
  found but has errors), 0 on failure. Added in version 6.

- **charsetconfigfromtext** (string)
CHAPTER 1. TEXIS WEB SCRIPT

Parses the given string buffer as a charset configuration, in the same format as a `charsetconfigfromfile` file. Returns 2 on success, 1 on partial success (file found but has errors), 0 on failure. Added in version 6.

- **charsetconverter**(string)
  The command and arguments to execute to convert character sets not known by the internal charset converter. The default (or if set empty) is the value set in `texis.ini` by [Texis] Charset Converter (p. 578). If that is unspecified, the value "${INSTALLDIR%/etc/iconv}" -f %CHARSETFROM% -t %CHARSETTO% -c is used. The variables ${INSTALLDIR%}, %CHARSETFROM% and %CHARSETTO% will be replaced with the Texis installation directory, source charset name and target charset name, respectively. Double quotes should be placed around single arguments that may contain spaces (e.g. the path to iconv) and will be removed in Unix versions. If the option %ALL% is given at the start, all charset conversions will be handled by this converter, even those that the internal converter knows. If the option %NONE% is given at the start (nothing else needed), no charset conversions will be handled by the converter; i.e. only the internal converter will be used. Returns previous setting. Added in version 5.00.1089408135 20040709. %NONE% added in version 7.02.1415897000 20141113.

- **charsetpartialconvok**(boolean)
  Whether to accept timeout/non-zero exit of external charset translator program, if at least some output was generated. Sometimes a few bad characters on a page can cause the translator to generate valid output for the rest of the page, but exit non-zero; if this setting is on, such partial output will be accepted. Added (and defaults to on) in version 5.01.1098470049 20041022. Returns previous setting.

- **charsetsrc**(string)
  Sets the character set to assume is the source for all pages fetched. Note that this forces the character set, i.e. all pages are interpreted as this character set even if labelled or detected differently. This setting should only be used as a last resort to force the character set for a mis-labelled or undetectable page; normally only `charsetsrcdefault` need be set. Note that this does not affect the charset for the output formatted text; see `charsettxt`. Currently recognized charsets are ISO-8859-1, UTF-8, UTF-16, UTF-16BE and UTF-16LE. Give empty string to set default, which is none/unknown – do not force charset, instead check next available source (i.e. charset set explicitly in page). Added in version 5.

- **charsetsrcdefault**(string)
  Sets the character set to assume for the source of pages fetched, when it is not forced (with `charsetsrc`), nor labelled nor detectable; i.e. a last-resort fallback. Note that this does not affect the charset for the output formatted text; see `charsettxt`. This setting is useful if most pages are correctly labelled, but a few are not labelled and Vortex is not correctly recognizing them. Give empty string to set default, which is none/unknown. Added in version 5.

- **charsettxt**(string)
  Sets the character set to return formatted text in (i.e. `<urlinfo text>`). The default in version 5 is UTF-8. If the charset of a given source page is different, its formatted text will be translated to this charset. The `charsettxt` value may be set to "source" or "src" to indicate that the source page charset should be used instead.\(^\text{32}\) It may be set to "" (empty string) to reset to the default value.

\(^\text{32}\)Note that even when `charsettxt` is set to "source", the page will still be internally translated to UTF-8 for processing.
• **del** (boolean)
  If true (default), the text within `<DEL>` blocks is included in the formatted text obtained with `<urlinfo text>`. If the del setting is false, this text is deleted. (This setting is the same as ignoredel but negated.) Added in version 3.01.962850000 20000705.

• **filedirrobotsfollow** (boolean)
  If true (default), file:// directory URLs’ HTML will contain a `<meta> robots tag value of follow`, which indicates to crawlers that the pages’ links should be followed. If false, the value will be nofollow. Added in version 5.01.1226709000 20081114.

• **filedirrobotsindex** (boolean)
  If true, file:// directory URLs’ HTML will contain a `<meta> robots tag value of index`, which indicates to crawlers that the pages’ content itself should be indexed. If false, the value will be noindex. The default is false, since directory contents are mostly filenames, which would clutter up the crawler’s index. Added in version 5.01.1226709000 20081114.

• **formatxmlashtml** (boolean)
  If true (default), XML documents are formatted and parsed as HTML (XSL stylesheets are not currently supported by the internal fetch formatter). If false, XML documents are left unparsed. Parsing XML as HTML will tend to return just the content of tags for formatted text, whereas leaving XML unparsed will return the entire raw document for formatted text. Added in version 5.01.1195086345 20071114.

• **formtxt or formtext** (boolean)
  Controls the select, input and textarea settings together. (This setting is the same as ignoreformtxt or ignoreformtext but negated.) Added in version 3.01.985900000 20010329.

• **ftpdirrobotsfollow** (boolean)
  If true (default), ftp:// directory URLs’ HTML will contain a `<meta> robots tag value of follow`, which indicates to crawlers that the pages’ links should be followed. If false, the value will be nofollow. Added in version 5.01.1226709000 20081114.

• **ftpdirrobotsindex** (boolean)
  If true, ftp:// directory URLs’ HTML will contain a `<meta> robots tag value of index`, which indicates to crawlers that the pages’ content itself should be indexed. If false, the value will be noindex. The default is false, since directory contents are mostly filenames, which would clutter up the crawler’s index. Added in version 5.01.1226709000 20081114.

• **input** (boolean)
  If true (default), the VALUE of `<INPUT TYPE=text>` tags is included in the formatted text obtained with `<urlinfo text>`. If the input setting is false, this text is deleted. (This setting is the same as ignoreinput but negated.) Added in version 3.01.985900000 20010329.

• **linelen** (integer)
  Sets the formatted-text line length to word-wrap at. The default is 75. Note that some lines may be longer that this, e.g. if word-wrap is disabled due to a `<PRE>` or similar tag. A value of 0 will set the

---

then translated back to the source charset.
CHAPTER 1. TEXIS WEB SCRIPT

default (i.e. 75). A value of -1 means infinite (no word wrap). Added in version 5.01.1119969728 20050628. Returns previous setting.

- **minclrdiff (integer)**
  The minimum foreground/background color difference that formatted text must have. If the color difference is less for a given section of text, the area will be blank instead.

  Sometimes extra padded keyword information – intended for web robots but not human users – is hidden in white-on-white text. This text is placed to artificially raise a page’s visibility in a search engine. However, since it often contains verbose or even completely off-topic keywords, such hidden text can be misleading. By setting `minclrdiff` this user-hidden text can be stripped from a Vortex-fetched page as well, and the resulting page ranked on its user-visible content only.

  The color difference is defined as: `\( \text{abs}(R_1 - R_2) + \text{abs}(G_1 - G_2) + \text{abs}(B_1 - B_2) \)`, where \( R_1 \), etc. are the RGB values for two colors. The default is 0, which implies that all text is included.

- **nestcomment (boolean)**
  Turn on or off nesting of HTML comments. With nesting on, comments may be nested. The default is off. Added in version 3.01.986950000 20010410. Aka `nestcomment/nestcomments/nestedcomments`.

- **select (boolean)**
  If true (default), the text within `<SELECT>` blocks is included in the formatted text obtainable with `<urlinfo text>`. If the `select` setting is false, this text is deleted. (This setting is the same as `ignoreselect` but negated.) Added in version 3.01.985900000 20010329.

- **showwidgets [add|del|set] [radio|select...] [all] [...]**
  Which `<form>` input widgets to display in formatted text, with square brackets or parentheses. Selected or checked widgets are further indicated with an asterisk. Displaying widgets can be useful to visualize where they are in relation to text, and which are selected/checked. The widgets to show are specified in one or more list arguments; they may be one or more of `button`, `checkbox`, `file`, `hidden`, `image`, `password`, `radio`, `reset`, `select`, `select-one`, `select-multiple`, `submit`, `text`, `textarea`, or `all` for all widgets. If the first token in the list is `add`, the widgets are added to the display list; if `del`, removed; if `set` (the default), the list is cleared and set to the specified list. The single widget `default` may also be specified (with `set`) to restore the default setting, which is to show no widgets. Added in version 5.01.1262085000 20091229.

- **strictcomment (boolean)**
  Turn on or off strict HTML comment parsing. With strict comments on, comments must start with “"!-"”, not just “!”; The default is on. Added in version 3.01.986950000 20010410. Aka `strictcomments`.

- **strike (boolean)**
  If true (default), the text within `<STRIKE>` blocks is included in the formatted text obtainable with `<urlinfo text>`. If the `strike` setting is false, this text is deleted. (This setting is the same as `ignorestrike` but negated.)

- **textarea (boolean)**
1.5. BUILTIN FUNCTIONS

If true (default), the text within `<TEXTAREA>` blocks is included in the formatted text obtained with `<urlinfo text>`. If the textarea setting is false, this text is deleted. (This setting is the same as `ignoretextarea` but negated.) Added in version 3.01.985900000 20010329.

• **xmltags** (boolean)
  Turns on or off interpretation of XML tags as tags. If on, tags that start with `<?` will be interpreted as an unknown HTML tag, i.e. suppressed from the formatted text. If off, such tags will be taken as text and will appear in the formatted text output. Added (and defaults to on) in version 5.01.1105303759 20050109. Returns previous setting.

• **utf8badencasiso88591** (boolean)
  If true (default), invalid bytes in UTF-8 source documents will be interpreted as ISO-8859-1 characters (and converted to `charsettxt`). If false, such bytes are replaced with question marks (“?”) as with other failed conversions. ISO-8859-1 is (erroneously) placed in UTF-8 text often enough that this assumption can generally be made. Note that such conversions still cause an error message and non-zero `<urlinfo errnum>`, however, unless `utf8badencasiso88591err` is set to false; this alerts the user to the erroneous document. Added in version 5.01.1244765000 20090611.

• **utf8badencasiso88591err** (boolean)
  If true (default), the interpretation of invalid UTF-8 bytes as ISO-8859-1 (when `utf8badencasiso88591` is true) still causes an error to be reported (if `charsetmsgs` true) and returned in `errnum`. If false, no error message is generated nor error returned. Note that if `utf8badencasiso88591` is false, `utf8badencasiso88591err` is ignored, as invalid UTF-8 bytes are then treated as any other failed conversion (mapped to question mark). Added in version 5.01.1244765000 20090611.

**Links**

The following `urlcp` settings control how links from formatted documents are processed:

• **baseurl** (string)
  The base URL to interpret relative links from, when making them absolute during formatting. The default (if empty) is to use the page URL or `<base href>` value. Added in version 7.05.1452546000 20160111.

• **contentlocationasbaseurl** (boolean)
  Whether to interpret the `Content-Location` header (if present) as the base URL for the document (can be overridden by `<base>` tag). The default is on. Added in version 5.01.1249455000 20090805.

• **eatlinkspace** (boolean)
  Whether to strip leading/trailing whitespace from links before processing into absolute links. The default is on. Added in version 3.01.968173351 20000905.

• **refs** (2, 4 or 6 list arguments)
  Which HTML tag/attributes are to be considered links, images, frames or iframes when formatting or reparenting HTML. Removing or adding tag/attributes can remove or add them from the returned lists.
of `<urlinfo links>` etc. This setting takes several parallel argument lists, in the form: `refs action flags... tag attr [attr2 val2]`:

- **action** (single value)
  A single value of `add`, `del`, or `set`, indicating how to apply the following arguments as a whole. If `add`, the arguments are added (flags ORed) to the existing values; if `del`, the arguments are deleted (flags cleared); if `set`, the existing values are cleared first and replaced with the arguments.

- **flags, ...** (list)
  Each value is a comma-separated list of one or more flags to apply:
  - `link` The tag’s `attr` value should be considered a link.
  - `image` The value should be considered an image.
  - `frame` The value should be considered a frame.
  - `iframe` The value should be considered an iframe.

  The above flags apply to both formatting (`<urlinfo links>`) and reparenting (p. 234). If `format` or `reparent` is appended to a flag, the flag applies only to that action instead.

- **tag** (list)
  The HTML tag referred to.

- **attr** (list)
  The attribute whose value(s) the flags apply to.

- **attr2** (list, optional)
  An optional second attribute that if specified, must be present and have the value `val2` for the flags to take effect. For example, by default `<INPUT SRC=...>` values are considered images, but only if the attribute `TYPE` is also present with the value `IMAGE`. This value can be empty or unspecified if not needed.

- **val2** (list, optional)
  Value for `attr2`. Required only if `attr2` value given.

The return value is the previous setting, as a single list with the `tag`, `attr`, `attr2` and `val2` arguments space-separated in each value. The return value may given as a single `refs set` argument to restore the previous settings. Also, the single argument `defaults` may be given to `refs set` to restore the built-in default settings. Note that HTML tags/attributes that are not currently known by the internal parser cannot be specified. Added in version 5.01.1159397148 20060927.

This example removes treating `<INPUT SRC=...TYPE=IMAGE>` values as images, and adds `<LINK SRC=...>` values as both images and links:

```<urlcp refs del image input src type image>
<urlcp refs add link,image link src>
```

- **scriptstrlinks** (string)
  Which types of JavaScript String links (those determined from scanning all JavaScript strings, instead of known true JavaScript links) to return. One or more of the values `none` (for no strings at all), `file` (for strings that resemble files), `protocol` (for strings that resemble URL protocols), or
1.5. **BUILTIN FUNCTIONS**

all (for all strings) may be specified. Note that script string links are unreliable and not guaranteed to be legitimate or even syntactically correct. This is a method of attempting to obtain links that the JavaScript module is otherwise missing. The strings are returned via `<urlinfo strlinks>` (p. 193). Returns previous setting. Added in version 5.00.1087588168 20040618. Default is protocol and file.

- **scriptstrlinkabs or scriptstrlinksabs (boolean)**
  Whether to absolute URLs from JavaScript String links. If on (the default) these URLs will be absolute. If off, they are left as-is (i.e. so the caller can perform additional scans or cleanup). Returns previous setting. Added in version 5.00.1087588646 20040618.

- **urlnonprint (string)**
  How to treat non-printable bytes (those outside the range `!` through `\009` inclusive) encountered in URL links of fetched pages:
  - **asis**
    Leave non-printable bytes alone.
  - **strip**
    Remove non-printable bytes.
  - **encode**
    Default: URL-encode non-printable bytes.

  Added in version 4.00.1006200000 20011119.

**Headers**

The following `urlcp` settings control what headers are sent with requests, which can affect what document the remote web server will return:

- **accept (2 arguments)**
  Set the HTTP `Accept` header list of acceptable/desired MIME types. Each value of the first argument ($value1) is a MIME media range, e.g. “text/html” or “image/*”. The corresponding value of $value2, if given, is a “quality” value, a percentage number from 0-100. If greater than 0, the q value of the corresponding media range is set to that value. If $value2 has fewer values than $value1, the last value of $value2, if any, is reused. See the HTTP specification for details on how these values are used by Web servers. The default `Accept` list (if not set) is “*/”*, e.g. any type.

Changing the `Accept` list may affect the content type of the document a Web server will send for a given URL, but it is no guarantee that the requested type(s) will be returned. It is up to the server to send the most appropriate form of a document based on the `Accept` list.

- **clearheaders (no arguments)**
  Clear any headers set with `header`.

- **header (list, 2 arguments)**
  Set the HTTP request headers given in the first argument, to the corresponding values in the second argument. This can be used to set additional headers not otherwise settable. Note that cookies are
automatically handled in version 4.01.1022000000 20020521 and later and thus Cookie headers do not generally need to be set in those versions.

In version 5.01.1245974000 20090625 and later, headers specified with this setting will replace builtin headers of the same name (e.g. Host etc.), instead of causing a second copy of the header to be sent. Note that setting/overriding builtin headers can cause erratic behavior, as user-specified values may interfere with library functionality. Builtin headers include Accept, Authorization, Connection, Content-Length, Content-Type, Cookie, Host, If-Modified-Since, Proxy-Authorization, Upgrade and User-Agent. All of these are set automatically by the library and/or have other <urlcp> settings that are the preferred method of controlling them.

Setting an empty value for a header will clear it (prevent it from being sent). It is not possible to send the same header two or more times: later values will merely replace earlier ones and the header will be sent at most once. To send multiple values for a header, set a single value with multiple tokens according to the HTTP syntax for the given header (typically comma-separated).

- **ifmodsince**(string)
  Sets the HTTP If-Modified-Since header to the given value. The argument is a time, either in Texis-parseable format or HTTP date format (www, dd mmm yyyy hh:mm:ss GMT). If the argument is empty, the header is cancelled.

  Setting the If-Modified-Since header creates a conditional request: the document is only returned if it has been changed since the given time, otherwise an empty document is returned. Setting this header on a per-page basis, to the Last-Modified value from the previous fetch, can reduce the traffic when re-walking a site: only new documents are returned. Note that it is up to the remote server to handle the If-Modified-Since header, and the given time is interpreted in its domain.

- **useragent**(string)
  Sets the User-Agent header sent with HTTP requests. The default is a Netscape-compatible string.

**JavaScript**

The following urlcp settings control JavaScript behavior. Note that most of these settings do not take effect unless the javascript setting is on:

- **javascript**(boolean)
  If on, JavaScript is enabled in documents: inline
  
  `<script language="JavaScript">...</script>` scripts in fetched documents are executed, which may affect the formatted text and links. Other document JavaScript actions are also enabled, as controlled by settings below. If off (default), JavaScript is disabled for documents (but may still be used for proxy auto-config, if PAC enabled; e.g.pacurl/pacscript p. 206). Returns previous value. Added in version 4.01.1023500000 20020607.

- **screenwidth**(integer)
  Sets the screen width as seen by JavaScript, i.e. the `window.screen.width` value. Default is 1280.

- **screenheight**(integer)
1.5. BUILTIN FUNCTIONS

Sets the screen height as seen by JavaScript, i.e. the window.screen.height value. Default is 1024.

- **scriptevents [add|del|set] [$events ...]**
  After a page is fetched, JavaScript events are triggered, and the appropriate event handler(s) called, which may add additional links to the result set of `<urlinfo links>`.
  This in effect simulates the user traversing a page and generating mouse-over, etc. events, which may collect some links that a “static” fetch and one-time run of the page wouldn’t, especially for certain JavaScript-navigated sites.

  The **scriptevents** setting controls which events are generated. Its argument(s) are a list of event names to trigger, from the following (default) list: Abort, AfterPrint, BeforePrint, BeforeUnload, Blur, Change, Click, Close, DblClick, DragDrop, Error, Focus, Help, KeyDown, KeyPress, KeyUp, Load, MouseDown, MouseMove, MouseOut, MouseOver, MouseUp, Move, Reset, Resize, Scroll, Select, Submit, Unload. The keyword <i>all</i> means all events. Events may also be named by handler, i.e. with the prefix <i>on</i>. If the first value of the first argument is <i>add</i>, the given events are added to the trigger list; if it is <i>del</i>, the events are deleted; if <i>set</i> (the default), the trigger list is cleared and set to the event list. Returns previous event list. Added in version 4.01.1023500000 20020607.

- **scriptlinks (boolean)**
  If true (default), all javascript: protocol links found on a page are executed (i.e. as if the user clicked on them). This may result in additional links being generated. (Such links often refer to page-specific functions, and thus must be run in the context of their referring page, not later on as an argument to `<fetch>`.) Note that the javascript protocol must also be enabled via the **protocols** setting. Returns previous setting. Added in version 4.01.1023500000 20020607.

- **scriptrealtimers (boolean)**
  Whether to run JavaScript timers (as set by setInterval() and setTimeout()) in real time. This is off by default, so that no real time is wasted waiting for the next timer (timers are still fired in proper order however). Returns previous value. Added in version 4.03.1050609000 20030417. See also **scriptmaxtimer** (p. 201).

- **scripturldecode (boolean)**
  Whether to URL-decode javascript:-protocol links before execution. On by default. Note that all links encountered in HTML are HTML-decoded first regardless. Returns previous value. Added in version 4.01.1024300000 20020617.

- **scriptvaryevents (boolean)**
  If true (default), after fetching a page, the options, checkboxes and radio buttons on a form are rotated through their values: each change generates additional JavaScript events, which may call JavaScript event handler(s), which may add additional links to the result set of `<urlinfo links>`. Note that this is in addition to the one-time call of event handlers controlled by **scriptevents**. Returns previous setting. Added in version 4.01.1025600000 20020702.

See also **getscripts** and **protocols** in the **Page Fetching** section (p. 202); **scriptmem**, **scripttimeout** and **scriptmaxtimer** in the **Resource Limits** section (p. 198); and **tracescript** and **scriptmsgs** in the **Informational/Trace** section (p. 228).
Cookies

The following `urlcp` settings control how cookies are processed by the fetch functions. The “cookie jar” is Vortex’s internal cache of cookies received during fetches and/or directly set by `cookiejar`. Starting with version 4.01.1022000000 20020521, Vortex automatically receives and sends cookies according to the RFC 2109 specification, with modifications to mimic popular browsers’ behavior. This largely eliminates the need for explicit header setting calls.

- `acceptnewcookies (boolean)`
  Whether to accept new cookies (i.e. first-time-seen). Returns previous setting. Added in version 5.01.1214274000 20080623. On by default.

- `acceptcookiemods (boolean)`
  Whether to accept cookie modifications, i.e. new non-empty non-expired values for cookies already seen. Returns previous setting. Added in version 5.01.1214274000 20080623. On by default.

- `acceptcookiedels aka acceptcookiedeletes (boolean)`
  Whether to accept cookie deletes, i.e. empty or expired values for cookies already seen. Returns previous setting. Added in version 5.01.1214274000 20080623. On by default.

- `acceptcookies (integer)`
  A deprecated alias for `acceptnewcookies`, `acceptcookiemods`, `acceptcookiedels` all together: bit 0 sets `acceptnewcookies`, bit 2 `acceptcookiemods`, bit 3 `acceptcookiedels`. Returns previous value. Note: Previous to version 5.01.1214274000 20080623, this was a boolean setting.

- `cookiedomainmatchself (boolean)`
  Whether a cookie’s Domain value should match itself when it has a leading dot, i.e. whether a Domain of “.x.y.com” should domain-match a host of “x.y.com” and thus be sent to that host. This is technically not permitted by RFC 2965, but is common browser behavior. On by default. Added in version 5.01.1225147420 20081027.

- `cookiejar $cookies [append]`
  Sets the “cookie jar” (internal list of cookies) by processing the text buffer `$cookies`. The format of the buffer is either Netscape or (in version 4.02.1042149825 20030109 and later) Microsoft Internet Explorer, so a browser cookie file can be directly read and processed to inherit a browser’s persistent cookies. The Netscape format is one cookie per line, with the tab-separated values: Domain IsOkAllDomain Path IsSecure IsHttpOnly Expires Name Value. E.g. the line:

  
  .site.com TRUE / FALSE FALSE 0 MyCookie MyValue

  would represent a session cookie named MyCookie with value MyValue sent for any path for any site in the domain .site.com. Note that the IsHttpOnly column was added in version 5.01.1244880000 20090613, and is not Netscape-format compatible.

  If the second argument is `append`, the cookie jar is not cleared before processing the buffer (the default is to clear the cookie jar first).
1.5. BUILTIN FUNCTIONS

- **cookies** (integer or boolean)
  Controls the `acceptnewcookies`, `sendcookies`, `acceptcookiemods` and `acceptcookiedels` settings together. If a boolean value is given, all settings are set to that value. If an integer is given, bit 0 applies to the `acceptnewcookies`, bit 1 to `sendcookies`, bit 2 to `acceptcookiemods` and bit 3 to `acceptcookiedels`. Returns the previous setting (integer). Added in version 4.01.1022000000 20020521. **Note:** values changed in version 5.01.1214274000 20080623.

- **cookiewildcards** (boolean)
  Controls whether wildcards are acceptable in cookies set with `cookiejar` (they are never acceptable from cookies seen “in the wild” from hosts). If true, the domain and/or path component of a `cookiejar` cookie may be a single asterisk (“*”) to indicate the cookie matches all hosts and/or all paths, respectively. The default is false, i.e. wildcards are not allowed. Added in version 5.01.1122648771 20050729 (default false in previous versions). Returns previous setting. Enabling wildcards can be useful when doing proxy cookie authentication, i.e. passing cookies sent to Vortex along to a `<fetch>`. The issue is that when cookies are sent back to a server (i.e. to Vortex), the original Set-Cookie header’s domain and path are lost, so they cannot be forwarded at the `cookiejar` call. Thus Vortex does not know the hosts/paths to forward the cookies to (and which not to). Enabling wildcards allows a domain/path of “*” to be set in `cookiejar`, which will forward the cookies to all hosts and paths, ensuring they reached the needed hosts. (It may also forward them to the wrong hosts, but that is unavoidable and usually benign.) Without wildcards, it is not feasible to set a domain/path syntax that encompasses all possible hosts and paths.

- **sendcookies** (boolean)
  Whether to send cookies (i.e. send the appropriate cookies from the cookie jar). On by default. Returns previous setting. Added in version 4.01.1022000000 20020521.

See also **cookiemsgs** in the Informational/Trace section (p. 228)

### Hostname Resolution

The following `urlcp` settings control DNS (hostname resolution):

- **dnsdomains or dnsdomain** (list)
  Takes a list of one or more search domains. Hostnames that do not contain a dot are searched for in each of these domains, in order. The default list is obtained from `/etc/resolv.conf` or the equivalent, depending on the system. Returns the previous list of domains. The limit is 5 domains. Has no effect if `dnsmode` is `sys`. Added in version 3.01.960500000 20000608.

- **dssignoretrunc or dnsignontc** (boolean)
  Sets whether to ignore the TC (truncated message) flag in DNS replies. This flag may be set by the name server if the reply is too large to fit in one UDP packet and was truncated; a typical cause of this is a hostname with many IP addresses, e.g. DNS-based load management. Since only one IP is used from the reply, it is safe to ignore this flag. On by default. Added in version 3.01.982400000 20010216.
• **dnsmode (string)**
  Sets the routines to use for domain name resolution. The second argument is either *int* to use the internal (non-blocking) Texis functions, *sys* to use the system’s C library functions (blocking), or (in version 3.01.993260000 20010622 and later), *intsys* which uses internal routines unless no nameservers can be found in which case the system C library functions are used. Returns the previous setting. The default is *intsys*. (Version 3.01.993260000 20010622 and earlier defaulted to *int* and had no *intsys* setting.) In version 4.04.1068200000 20031106 and later, an empty string will restore the default setting.

  Note that the *sys* (and *intsys* if no nameservers found) routines are blocking: thus a PARALLEL fetch or nslookup will in effect be serial, and slow DNS traffic may even cause the timeout to be exceeded for some fetches. Also, certain information and features of nslookup and nsinfo may be ignored if *sys* mode is set, as well as other DNS-related settings in *urlcp*.

• **dnsrecurse (boolean)**
  Sets whether to request recursion to other nameservers when resolving hostnames. The default is on. Added in version 3.01.989630000 20010511. Returns previous setting.

• **dnsretrans (integer)**
  Sets the initial retransmit interval (in seconds) for DNS lookups. Has no effect if *dnsmode* is *sys*. Default 5.

• **dnsretry (integer)**
  Sets the max number of retries for DNS lookups. Returns previous setting. Has no effect if *dnsmode* is *sys*. Default 4.

• **dnsservices (list)**
  Takes a list of one or more of the strings *hosts* or *bind*, indicating the name resolution services to use and the order in which to try them. Returns previous setting. Has no effect if *dnsmode* is *sys*.

• **nameservers (list)**
  Sets the list of nameservers to use, in order, for DNS resolution. Takes a list of dotted-decimal IP address strings, and returns the previous list of nameservers. Has no effect if *dnsmode* is *sys*.

### Informational/Trace

The following *urlcp* settings control various informational and trace behaviors. Note that increasing these settings may cause sensitive information to be printed and/or sent to vortex.log, e.g. authentication or password data.

• **badhdrmsgs or badheadermsgs (boolean)**
  Whether to issue a putmsg for malformed response headers. Added (and defaults to on) in version 5.01.1102538511 20041208. Returns previous setting.

• **charsetmsgs (boolean)**
  Whether to issue a putmsg for charset conversion problems. The default is on. Added in version 5.00.1089408135 20040709. Returns previous value.
• **cookiemsq** (boolean)
  If true, additional messages are printed when invalid cookies are received and discarded, via the putmsg mechanism (p. 581). If false (default), invalid cookies are silently discarded. Returns previous setting. Added in version 4.01.1022000000 20020521. See also the Cookies section (p. 226).

• **fetchmeter** (boolean or string)
  Whether to print a progress meter during fetches. If true, a hash-mark progress meter is printed for uploads and downloads, where possible. The setting can also be a meter type: one of “simple”, “percent” or “none”. A true value enables a meter of type “simple”. Returns previous value. Added in version 6.

• **pacmsgs** (boolean)
  Whether to print PAC (proxy auto-config) error messages, e.g. from JavaScript and/or parsing. The default is on. Added in version 7.05.

• **putmsg** pass|save|all|clear [on|off [on|off ...]] (boolean)
  Controls the disposition of putmsgs from fetches. The first argument is a boolean action, which is set to the corresponding boolean value from the second (or later) value argument:

  - **pass**: Whether to pass fetch-related messages through to the normal putmsg mechanisms, i.e. printing, logging and/or <putmsg> function (if enabled). Since messages can also be saved for later retrieval via <urlinfo>, this is often turned off to prevent the need for duplicate handlers. Non-fetch messages (<sql>, etc) are unaffected by this. On by default.

  - **save**: Whether to save messages in a buffer for later retrieval with <urlinfo putmsgs>. Calling <urlinfo putmsgs> inside a <fetch parallel> loop will return only the messages for the just-completed fetch, making disambiguation of messages easier than with the normal script <putmsg> function callback mechanism. On by default.

  - **all**: Set pass and save (in that order) according to the next value argument values.

  - **clear**: Clear the current fetch putmsg buffer. No arguments.

Returns the previous setting(s) (two values if all used). The putmsg setting was added in version 6.

• **redirm</setlist>ngs** (boolean)
  If true, info messages about redirects will be printed (via the putmsg mechanism). If false (default), such messages are suppressed.

• **scriptmsgs** (boolean or int)
  If true, JavaScript errors are printed/logged via the putmsg mechanism (p. 581). If false (default), such errors are suppressed. Many JavaScript pages generate errors due to script bugs, or varying execution environments, so this setting is off by default. Turning it on can help isolate JavaScript-related issues on a page. Note that this applies to syntax-type errors from the engine; certain external fatal errors are still reported regardless, such as exceeding resource limits (time/memory/page size). Returns previous setting. Added in version 4.01.1023500000 20020607.

In version 7.01.1385154000 20131122 and later, an integer value is interpreted as a set of bit flags controlling which types of errors to report: flag 0x1 for ordinary script errors, flag 0x2 for assert() failures (which were unreported prior to version 7.00.1372360000 20130627). String boolean values now set all flags on or off.
• **traceauth** (integer)
  Set trace message level for client/server authentication calls. Value is a bitwise OR of any of the following flag values; the default is 0 (i.e. no tracing):
  
  - 0x0001: Protection spaces made (added, expanded, deleted etc.)
  - 0x0002: Protection spaces used (attached, detached)
  - 0x0004: Authentication objects used (stop, start, handle, connection closed, headers etc.)
  - 0x0008: Authentication protocol-specific messages
  - 0x0010: Keep-Alive (persistent connections) messages
  - 0x0020: WWW-Authenticate parsing
  - 0x0040: Connection open/close/idle/re-use
  - 0x0080: Socket/descriptor/handle open/close

  *Note:* Setting a non-zero value may also print sensitive data such as authentication or password strings. Flag values may be added to or change in future releases. Added in version 5.1. Note that `traceauth` values changed in version 7.06.1465245000 20160606.

• **tracecalls** (integer)
  Set trace message level for fetch-related function calls. This is a bitwise OR of integer values indicating which calls to issue messages for; the default is 0 (i.e. no tracing):
  
  - 0x0001: `<urlcp>` calls
  - 0x0002: `<fetch>` calls
  - 0x0100: Print full arguments (do not truncate)

  Note that these values may change in future releases, as this is a debug setting. Added in version 5.01.1200000000 20080110.

• **tracedns** (integer)
  Set trace message level for DNS (hostname lookup) calls. Value is an incrementing integer, i.e. each level provides messages at its level and all preceding levels; the default is 0 (i.e. no tracing):
  
  - 0 No tracing
  - 1 `gethostbyname`, `gethostbyaddr` calls; miscellaneous warnings
  - 2 Service calls (hosts/BIND/NIS); `nsswitch.conf` and `host.conf` parse
  - 3 Packets sent/received
  - 4 Individual config entries, `/etc/hosts` entities, reply RR entries
  - 5 All socket calls

  Values may change or be added to in future releases. Added in version 3.0.948765000 20000124.

• **traceencoding** (integer)
  Sets trace message level for content and transfer encoding calls. Value is a bitwise OR of any of the following flags; the default is 0 (i.e. no tracing):
  
  - 0x0001: Open/close objects
1.5. BUILTIN FUNCTIONS

- 0x0002: ... TranslateEncoding filter calls
- 0x0004: Decoder object state changes etc.
- 0x0040: Data read
- 0x0080: Data written

Values may change or be added to in future releases. Added in version 5.01.1249039000 20090731.

• tracefetch (integer)
  Set trace message level for fetch library calls. Value is a bitwise OR of any of the following flags; the default is 0 (i.e. no tracing):

  - 0x0001: Top-level (user-initiated) fetch
  - 0x0002: Sub-object (script/frame) fetch
  - 0x0004: Sub-object fetch from cache
  - 0x0008: Static (HTML) links added while formatting
  - 0x0010: Script links added
  - 0x0020: Static images/frames/etc. added
  - 0x0040: Script images/frames/etc. added
  - 0x0080: Top-level fetch from user-provided source
  - 0x0100: Proxy cache activity (except script fetches)
  - 0x0200: Redirect fetch
  - 0x0400: Authorization fetch
  - 0x0800: Proxy retry fetch (attempt with 2nd+ proxy)
  - 0x1000: Empty-response retry fetch
  - 0x2000: PAC script fetch

  Added in version 5.01.1147127000 20060508. Bit 0x0080 added in version 6.00.1308203000 20110616. Bit 0x0100 added in version 7.04.1446832000 20151106. Bits 0x200 - 0x2000 added in version 7.06.1471876000 20160822. Setting 0x3e03 should trace all network activity. Values may change or be added to in future releases.

• tracescript (integer)
  Set trace message level for JavaScript calls. Value is a bitwise OR of any of the following flags; the default is 0 (i.e. no tracing):

  - 0x0001: Inline scripts run
  - 0x0002: Remote scripts run
  - 0x0004: javascript: -protocol links run
  - 0x0008: Timers run
  - 0x0010: Objects checked for events
  - 0x0020: Modify-and-call-event-handler objects run
  - 0x0040: Event handlers run (normal or modify)
– 0x0080: Check-string-for-URL called
– 0x0100: Check-string-for-URL link added
– 0x0200: Script output inserted (into raw HTML parse buffer)
– 0x0400: Event handlers added
– 0x0800: Memory allocated
– 0x1000: Memory freed

Added in version 4.04.1082500000 20040420. Values may change or be added to in future releases.

• traceskt (integer)
  Set trace message level for socket (i.e. over-the-wire network traffic) calls. Value is a bitwise OR of any of the following flag values; the default is 0 (i.e. no tracing):

  – 0x0000001: After open(), close(), accept() etc.
  – 0x0000002: After select()
  – 0x0000004: After reads
  – 0x0000008: After writes
  – 0x0000010: After ioctl(), getsockopt()
  – 0x0000020: Currently unused
  – 0x0000040: The data read
  – 0x0000080: The data written, or sent with ioctl()
  – 0x0000100: Details of SSL verification of peer certificate
  – 0x0010000: Before open(), close(), accept() etc.
  – 0x0020000: Before select()
  – 0x0040000: Before reads
  – 0x0080000: Before writes
  – 0x0100000: Before ioctl(), getsockopt()
  – 0x0200000: Currently unused
  – 0x0400000: The data buffer before reads
  – 0x0800000: The data buffer before writes or ioctl()
  – 0x1000000: Request SSL peer certificate (if not already requested)

Note: Setting a non-zero value may also print sensitive data such as authentication or password strings. Added in version 5.01.1103682324 20041221. Values may be added to or change in future releases.

• verbose (integer)
  Sets the verbosity level; used for debugging. This is a bitwise OR of integer values indicating what network protocol information to print:

  – 1: Responses read
  – 2: Requests sent
1.5. BUILTIN FUNCTIONS

- 4: Headers read
- 8: Headers sent

Note: Setting a non-zero verbosity level may also print sensitive data such as authentication or password strings. Previous to version 5.01.1093600000 20040826 the verbose setting was an incrementing integer, 0 to 2.

Miscellaneous

The following urlcp settings control miscellaneous page fetching behaviors:

- **allowbadchunkedinfo**(boolean)
  If on (the default), try to allow certain bad chunked Transfer-Encoding information if encountered in a response (e.g. missing chunk size), by just passing through remaining data as-is (because chunked coding is mostly clear text anyway). If off, fail the transaction. Note that regardless of the setting’s value, bad chunked information may indicate corrupt data in the response. May help recover a fetch when the server erroneously reports chunked coding in the response. Added in version 6.00.1315620000 20110909; previous versions behaved as if setting was off.

- **alarmclose**(boolean)
  Whether to use an alarm() to terminate a blocking connection close(). The default is off. Added in version 4.04.1050700000 20030418.

- **badhdrmidok** or **badheadermidok**(boolean)
  Whether to accept malformed headers in the middle (i.e. not last) of the response headers. If on, malformed headers will be discarded if followed by at least one valid header. If off, or no valid header follows, the malformed header will be considered the start of the body (which it might very well be). Added (and defaults to on) in version 5.01.1102538511 20041208. Returns previous setting.

- **checkidleconneof**(boolean)
  If on (the default), idle connections in the Keep-Alive cache are checked for EOF (i.e. server closure) before being reused. The server may have timed out the connection while the socket was idle, which would otherwise cause the next fetch to fail. Added in version 5.01.1115130972 20050503 (default off in previous versions). Returns previous setting.

- **clearproxycache**(no arguments)
  Clears the cache of “bad” (non-responsive) proxies, which are set to lower priority in PAC responses when other proxies are listed (see proxyretrypretrydelay, p. 200). Also clears the last PAC fetch timestamp (see pacfetchretrypretrydelay, p. 200). Added in version 7.05.

- **closeidleconn**(no arguments)
  Closes any currently idle connections in the Keep-Alive cache. Returns 1 if successful, 0 on error. Added in version 5.00.1093895662 20040830.

- **defaults**(no arguments)
  Resets all urlcp settings to their default values.
• **delaysave** (boolean)
  Sets whether to delay the saving of output when using `<submit TOFILE=$file>` until the connection starts returning data. The default is off, e.g. open the file immediately (always deleting previous copy). Turning this setting on is useful when repeatedly downloading to the same file, e.g. obtaining a periodic update of a large data file. The original file will then be preserved if the new fetch fails immediately (e.g. remote server down), yet saves the disk space of a separate backup copy. Added in version 4.0.997840000 20010814.

• **domvalue** $dom $value
  Sets the value of the DOM item indicated by $dom to $value. Note that this does not affect the JavaScript DOM, but the near-parallel page DOM. This can be used to set form input values, etc. and then obtain the submit URL and content via `<urlinfo domvalue>`. Added in version 5.

• **emptyhttp09ok** (boolean)
  Whether to accept empty HTTP/0.9 responses, i.e. a 0-byte response with no headers. Such responses are technically legal (an empty HTTP/0.9 document), but since few pre-HTTP/1.0 servers exist, are more likely indicative of a server error. If off, an error message is issued and an error is set. Added (and defaults to off) in version 5.01.1097502096 20041011. Returns previous setting.

• **linger** (boolean)
  Sets whether to set SO_LINGER time of 4 seconds on sockets. Added (and defaults to off) in version 5.01.1105153893 20050107. Returns previous setting.

• **reparent** (string)
  If given a full path (e.g. “/local/tree”), sets reroot reparent mode and uses that path as the local tree root. If given a full URL (e.g. “http://somesite.com/dir/page.html”), sets abs reparent mode and uses that URL as the page’s URL (this is not recommended; links may become incorrect).

• **reparentimg** (boolean)
  If true (default), image links will be reparented; if false, they will not. Only significant if reparentmode is not off.

• **reparentmode** (string)
  The returned HTML from a page will be reparented: all the links will be changed in the raw document returned. How the links are modified depends on the mode:
  
  - **abs** or 1
    Make all links absolute. With this mode, a page can be fetched from a remote site and the returned document placed directly in a local source tree, and even relative links will correctly point to the original locations. If a URL is set with reparent, the page is reparented as if it were fetched from there, instead of its actual location (the default). (Setting a URL is not recommended, as the links may become incorrect.)
  
  - **reroot** or 2
    Re-path same-site links as if the entire remote site were being copied locally to a subtree rooted at the URL path given with reparent. For example, with a reparent path of `/local/tree`, if the URL `http://somesite.com/dir/page.html` is fetched, it is assumed it will be saved to `/local/tree/dir/page.html`. Thus a link such as `/top/list.html` will become `/local/tree/top/list.html`. The link
../upone.html would become /local/tree/upone.html. If no path is set with re-parent, links become relative, as if the root were /.

- mirror or 3
  Make all links absolute, URL-encode them, and prefix the re-parent URL.

- relatedfiles or 4
  For an email message, change all internal links (to other parts of the message) to their safe filenames, as same-dir relative links. All other (external) links will be made absolute. This mode is used internally by the mimeEntityGetBody() function (p. 513) when re-parenting. Note: since it requires additional parsed email message information, it cannot currently explicitly be used by Vortex scripts.

- hideexternal or 5
  Change all external links – those referring to outside the page’s current directory or below – to have a prefix of “thismessage:”, to prevent their access. This can be used to hide/disable external references in HTML email message bodies during web display, after the HTML has been message-reparented by the mimeEntityGetBody() (p. 513) function.

- off or 0
  Turn off all re-parenting. Added in version 3.01.968705387 20000911. Default.

Note that the re-parent and re-parentmode settings do not affect the links returned by <urlinfo links>.

- sendemptycontent (boolean)
  Whether to set Content-Length: 0 for empty requests. Some servers will time out, expecting an EOF from the client, if an empty request is sent with no Content-Length. Added (and defaults to on) in version 5.01.1097006042 20041005. Returns previous setting.

- shutdownwr (boolean)
  Turns on or off the use of shutdown (SHUT_WR) on HTTP or Gopher sockets when all data has been sent and the connection is not to be re-used (e.g. Keep-Alive has expired or is not in use). This sends an EOF to the server to indicate that the client has finished sending data. Some broken servers may expect such an EOF even if Content-Length is set properly in the request, and may thus time out the request waiting for one. (Note that for shutdown() to actually be used, it may be necessary to disable Keep-Alive via <urlcp maxconnrequests 1>.) Added (and defaults to on) in version 5.01.1105300267 20050109. Returns previous setting.

- urlcanonslash (boolean)
  Whether to canonicalize backslashes (“\”) to forward slashes (“/”) in URLs. On by default. Turning off may impair URL parsing.

- urlcollapseslashes (boolean)
  Whether to collapse multiple forward slashes (e.g. “//”) to a single forward slash in the path part of URLs. Note: does not affect the double-slash that immediately follows the protocol plus colon in some URL protocols. Off by default. Added in version 7.03.1434400000 20150615.

DIAGNOSTICS

tool returns 1 on success, or 0 or nothing on error, except as noted under specific options.
EXAMPLE

<urlcp "maxpgsize" "1MB">  <urlcp "timeout" 300>
<fetch "http://www.somesite.com/bigpage.html">

CAVEATS

The urlcp function was added Mar. 26 1997. Various settings were added later.

SEE ALSO

fetch, submit, urlinfo, nslookup
1.5.41 urlutil – URL/network utility

SYNOPSIS

<urlutil $action [$arg ...]>

DESCRIPTION

The urlutil function provides URL and other network-related utility functions. The $action argument determines what it does:

- abs $absurl $relurl or absurl $absurl $relurl
  Makes URLs absolute (fully specified). The $absurl values are one or more absolute page URLs. The $relurl values are corresponding links – relative or not – from those page(s). For each $relurl value, its absolute value is returned, as if it were a link on that page. If there are fewer $absurl values than $relurl values, the last $absurl value is re-used.

- charsetcanon $charset
  Returns canonical name for charset name $charset, according to current Charset Config file (p. 578). Can be used to map charset aliases to canonical names.

- charsetconv $buf $from [$to]
  Converts text buffer $buf from charset $from to charset $to. The default for $to if unspecified or empty is the current <urlcp charsettxt> setting. Some character sets may require the use of an external charset converter (the default is iconv, see <urlcp charsetconverter> to change it), which is automatically executed when needed. Added in version 5.00.1090598954 20040723.

- charsetdetect $buf
  Returns guess at charset for text buffer $buf, or “Unknown” if charset unknown. Only limited charset detection is supported, primarily UTF-8, UTF-16BE/UTF-16LE, and all-7-bit ISO-8859-1. Added in version 7.02.1398457000 20140425.

- filepath $u
  Takes $u, which must be a file:// URL, and returns the local file path that would be used to read the file, as determined by the current <urlcp fileroot> etc. settings.

- pacinit
  Initializes proxy auto-config by fetching PAC script (if configured, p. 206) and running it. Returns 1 if successful, 0 if not. The error from the fetch, messages from the fetch and script execution, and the body of the script (if fetched) are available afterwards via <urlinfo>. If no PAC script nor URL is configured, or the script was already initialized, no action is taken, and 1 (success) is returned.

Calling <urlutil pacinit> when using a PAC script is not necessary: the PAC script is automatically fetched and run when needed, i.e. at the first <fetch> or <submit>, and any messages at PAC initialization are reported. However any PAC failure during such automatic initialization merely translates into a Proxy auto-config error for the <fetch>. The <urlutil pacinit> action provides a way to get more detailed information about the PAC script, if desired for diagnostic purposes.
• split $u $parts
  Splits a URL into one or more parts. The $u value is the URL to split. The $parts values are a list of the parts to return, in the same order, as values in $ret. The parts can be protocol, user, pass, host, port, path, type, query or anchor.

• sslcertificate $pem tostring
  Parses an SSL certificate string buffer $pem (in PEM format). The tostring sub-action returns a human-readable string version of the certificate, with subject, issuer, expiration etc. printed. This can be used to view a server certificate returned from <urlinfo sslservercertificate>.

Several actions take inet type arguments, which are IP network and/or host address specification strings of the form: $N [ . N [ . N [ . N ] ] ] / ( B | : I P )$ where $N$ is a decimal, octal or hexadecimal integer from 0 to 255, $B$ is a decimal, octal or hexadecimal netmask integer from 0 to 32, and $IP$ is an IP address netmask of the form $N [ . N [ . N [ . N ] ] ]$. If only $x$ $Ns are specified, the last $N$ may be $5 – x$ bytes in size instead of 1 byte. E.g. “1.2.65535” is legal (last $N$ is 2 bytes), whereas “1.2.3.65535” is not. If no netmask ($/B|:IP$) is specified, the netmask will be calculated from standard class A/B/C/D/E rules, but it will be at least large enough to include all specified bytes of the IP. (Thus, to get the class A/B/C/D/E netmask of an IP address via inetnetmask, just give the first/highest $N$ byte of the IP, as this is the sole determiner of class.) If an IP netmask is specified, only the largest contiguous set of most-significant 1 bits are used.

Examples: “1.2.3.4” (netmask is /32 because 4 bytes given), “10” (netmask is /8 because it is a Class A address), “1.2.3.4/10”, “67305985” (e.g. 4.3.2.1).

The inet actions were added in version 5.01.1112986377 20050408, and include the following (see also the SQL equivalents):

• inetabbrev $inet
  Returns the shortest representation of $inet. This will include all contiguous most-significant bytes of the network, non-zero bytes of the host, and a netmask. Other trailing 0 bytes of the IP may be trimmed. Empty string is returned on error.

• inetcanon $inet
  Returns canonical representation of $inet. This is dotted-decimal with all 4 bytes, and a /$N$ netmask. Empty string is returned on error.

• inetnetwork $inet
  Returns 4-decimal IP address with the network bits of $inet, and the host bits set to 0. Empty string is returned on error.

• inethost $inet
  Returns 4-decimal IP address with the host bits of $inet, and the network bits set to 0. Empty string is returned on error.

• inetbroadcast $inet
  Returns 4-decimal IP broadcast address for $inet, i.e. with the network bits, and host bits set to 1. Empty string is returned on error.

• inetnetmask $inet
  Returns 4-decimal IP netmask for $inet, i.e. with the network bits set to 1, and host bits set to 0. Empty string is returned on error.
• **inetnetmasklen $inet**
  Returns integer netmask length of $inet. -1 is returned on error.

• **inetcontains $inetA $inetB**
  Returns 1 if $inetA contains $inetB, i.e. every address in $inetB occurs within the $inetA network. 0 is returned if not, or -1 on error.

• **inetclass $inet**
  Returns class of $inet, e.g. A, B, C, D, E or classless if a different netmask is used. Empty string is returned on error.

• **inet2int $inet**
  Returns integer representation of IP network/host bits of $inet (i.e. without netmask). Useful for compact storage of IPv4 address as integers instead of strings. -1 is returned on error (note that -1 may also be legitimately returned for an all-ones IP address, e.g. “255.255.255.255”).

• **int2inet $i**
  Returns inet string for integer $i taken as an IP address. Since no netmask can be stored in the integer form of an IP address, the returned IP string will not have a netmask. Empty string is returned on error.

**EXAMPLE**

```bash
<urlutil abs "http://example.com/dir/page.html" "other.html">
```

The return value in $ret would be “http://example.com/dir/other.html”.

**CAVEATS**

The `urlutil` function was added in version 3.0.957600000 20000505.

**SEE ALSO**

`fetch`, `urlinfo`
1.5.42 nslookup – domain name and IP address lookup

SYNOPSIS

<nslookup [options] $host|$ip[ /]> 

or 

<nslookup PARALLEL[=n] [opts] [HOSTS|ADDRS=]$hosts|$ips [$loopvar ...]> 

DESCRIPTION

The nslookup function can resolve a hostname into an IP address and vice versa. By default, it looks up a hostname by name, returning the dotted-decimal IP address corresponding to the host. If given an IP address, however, it will perform a reverse lookup and return the host name for the given IP address.

With the second syntax (the PARALLEL flag), name resolution can occur in parallel, with multiple names resolving simultaneously. Behavior is similar to <fetch PARALLEL> (p. 179), in that n (default all) of the names are resolved at a time, and the results are looped over inside the <nslookup> block, as they are completed. Additional variables, if given after the host names/IP addresses, can be looped over in the same return sequence. $loop and $next are set inside the loop as in fetch.

Options that can be set are:

- BYADDR
  Forces all lookups by address, i.e. IP-to-name translation. If a given value doesn’t look like an IP address, it is assumed to be a host name already resolved, and is returned as-is. Overrides the HOSTS=/ADDRS= implied setting.

- BYNAME
  Forces all lookups by name, i.e. name-to-IP translation. If a given value looks like an IP address, it is assumed to be already translated and is returned as-is. Overrides the HOSTS=/ADDRS= implied setting.

- MTERR
  Return empty string if name or IP address cannot be resolved. By default, a given value is returned as-is if the lookup fails, so that a bulk translation of addresses, e.g. from a web log, doesn’t have to be checked individually for errors when printing results.

- HOSTS=$hosts or ADDR=$ips
  Sets the list of names to be looked up in parallel. Either by-name or by-address translation is forced accordingly; but the BYNAME/BYADDR flags override this. If just a list of names is given without HOSTS=/ADDRS= or the BYNAME/BYADDR flags, by-name or by-address translation happens on a per-value basis depending on whether the name looks like an IP address or not.
After each value is resolved, more information about the lookup can be obtained with the nsinfo function (p. 242), such as the list of aliases (if any), or additional IP addresses. As with all looping statements, <BREAK> can be used inside a looping nslookup to stop pending lookups and end the loop. The urlcp function (p. 198) is used to control various aspects of nslookup behavior.

DIAGNOSTICS

dslookup returns the resolved IP address or domain name, as appropriate. On error, either the original value or an empty string is returned, depending on the MTERR flag.

EXAMPLE

This example prints the distinct hostnames of the last 100 clients to access the web server. It does this by reading the last 100 lines of the transfer log and pulling out the IP addresses, then using nslookup to resolve the names. Note that the list is uniq’d before being passed to nslookup, to avoid duplicate lookups of the same name. Also, the names are resolved only 3 at a time to save traffic on the nameserver(s). Because of the BYADDR flag, any addresses that are already resolved in the log file (i.e. if name lookups were turned on in the server) are passed through as-is:

```plaintext
<READLN REV /usr/local/morph3/transfer.log></READLN>
<rex "">=[^\s]+" $ret> !-- get the IPs -->
<uniq $ret ICASE><ip = $ret> !-- remove duplicates -->
<nslookup PARALLEL=3 BYADDR $ip> !-- look up 3 at a time -->
$ret
</nslookup>
```

By turning off reverse name lookup in the web server and using a script such as this to resolve names only as needed, network traffic can be decreased and web server response time increased.

CAVEATS

The nslookup function was added in version 3.0.951800000 20000228.

As with fetch PARALLEL, it is easy to overload a server with too many requests at once. This is even more true with nslookup, because all those requests are going through the same local nameserver even for different-domain hosts. Use caution with the PARALLEL flag; always given a small number.

Resolved names are not cached, so multiple lookups of the same name or IP (e.g. from a web log) are discouraged; use uniq or the like to avoid duplicate lookups and save traffic.

If <urlcp dnsmode sys> is set, lookups are serial even if PARALLEL is set, due to C lib constraints.

SEE ALSO

urlcp, nsinfo, fetch
1.5.43 nsinfo – get info from last name lookup

SYNOPSIS

<nsinfo $name [$which]>

DESCRIPTION

The nsinfo function obtains information about the most recently completed nslookup (p. 240). It can be called after a single nslookup or within an nslookup loop. The $name argument is one of the following:

- **addr**
  Returns the list of IP addresses given in the reply.

- **name**
  Returns the canonical host name.

- **aliases**
  Returns any aliases known for the host.

- **errnum**
  Returns the error code (not a protocol code) from the last lookup. 0 indicates a successful reply. See below for other numbers.

- **errmsg**
  Returns the error message from the last lookup. The possible errnum codes and their corresponding errmsg messages are:

  0 Ok
  1 No recovery
  2 Try again
  3 Cannot open hosts file
  4 Cannot connect to nameserver
  5 Nameserver connection refused
  6 Nameserver query timeout
  7 Host not found
  8 Bad IP address
  9 Incorrect usage
  10 Out of memory
  11 Bad service
  12 No data
1.5. BUILTIN FUNCTIONS

13 Internal error
14 Unknown error

- replyheader
  Returns text information about the reply header, including id, flags, opcode, and response code. Added in version 3.01.989630000 20010511.

- replyrecords
  Returns text information about the reply's records. An optional second argument can be one of query, answer, authority, additional or all to return records from just a particular section (or all sections). Each value returned is a space-separated list of name, TTL, class, type, and data (answer). Added in version 3.01.989630000 20010511.

- rawreply
  Returns the raw packet data from the last reply, as a varbyte field. Largely superseded by replyheader and replyrecords options.

DIAGNOSTICS

nsinfo returns the requested information from the last nslookup completed.

EXAMPLE

<nslookup $host>
IP address: $ret
<nsinfo aliases>
Aliases: <LOOP $ret> $ret </LOOP>

CAVEATS

The nsinfo function was added in version 3.0.951800000 20000228.

If <urlcp dnsmode sys> is set, less information may be available from the reply.

SEE ALSO
urlcp, nslookup
1.5.44 options – print options list

SYNOPSIS

<options $values $selected $output
   [$classes [$ids [$titles]]]>

DESCRIPTION

The options function prints an HTML <option> tag for each value in $output. The HTML value attribute is set to the corresponding value of $values. If it matches a value in $selected, then the HTML selected attribute is set. Finally, the value of $output is printed as the user-visible value of the tag.

If there are fewer values of $values than $output, the remaining options do not have value attributes set (and the $output value is checked instead for the selected attribute). Thus, a no-values $values argument would cause no value attributes at all to be set. This can be used to minimize the amount of HTML text sent for a large list of small values. Note that if $values is a single empty-string value, it is also treated as an empty list (no values); this allows a literal empty string to be passed to indicate “no value attributes”. (To ensure value attributes are always set, make sure $values has at least two values, even if empty.)

In version 5.01.1219783000 20080826 and later, the output is XHTML 1.0 Strict compliant, and the optional $classes and $ids arguments may be given. The $classes argument is a parallel list of class attribute values to set; if the list is short, the last value is re-used. If no values (or a single empty value) is given, no class attributes are set.

The $ids argument is a parallel list of HTML id attribute values to set; if the list is short, the last value is re-used, but with its last numeric section incremented each time (to make it unique). This allows control of the starting id value, with the convenience of the function making the rest of the sequence unique. If no values (or a single empty value) is given, no ids are printed. Empty $ids values in the middle of the list will be printed as “option” with an incrementing number.

In version 5.01.1225758000 20081103 and later, an optional $titles argument may be given. If non-empty, each value specifies the title attribute to set for the corresponding item.

DIAGNOSTICS

options returns nothing.

EXAMPLE

<$values = red green yellow blue>
<$selected = green>
<select name="color">
  <options $values $selected $values>
</select>

CAVEATS

The options function was added Sep. 20 1996. The $classes and $ids arguments were added in version 5.01.1219783000 20080826.
All values are HTML-escaped when printed.

SEE ALSO

radiobutton, checkbox
1.5.45  radiobutton – print radio buttons list

SYNOPSIS

<radiobutton $name $values $selected $output
   [$classes [$ids [$titles]]]

DESCRIPTION

The radiobutton function generates a list of radio buttons (HTML <input type="radio"> tags), one for each value of $output. The $name list is the name of the tag; if it has fewer values than $output the last name is re-used. The $values list is the list of value attribute values. Values that match a value of $selected are set checked. The $output list is the list of user-visible values. If $values has fewer values than $output, the corresponding $output values are used.

In version 5.01.1219783000 20080826 and later, the output is XHTML 1.0 Strict compliant, and the optional $classes and $ids arguments may be given. The $classes argument is a parallel list of class attribute values to set; if the list is short, the last value is re-used. If no values (or a single empty value) is given, no class attributes are set.

The $ids argument is a parallel list of HTML id attribute values to set; if the list is short, the last value is re-used, but with its last numeric section incremented each time (to make it unique). This allows control of the starting id value, with the convenience of the function making the rest of the sequence unique. If no values (or a single empty value) is given, the ids are named after $name with an incrementing number.

In version 5.01.1225758000 20081103 and later, an optional $titles argument may be given. If non-empty, each value specifies the title attribute to set for the corresponding item.

Each $output value is printed in a <label> associated with its radiobutton via the id, so that clicking on the displayed text also toggles the radiobutton, making it easier to click.

DIAGNOSTICS

radiobutton returns nothing.

EXAMPLE

<${values = red green yellow blue}><${selected = green}>
<radiobutton "color" $values $selected $values>

CAVEATS

The radiobutton function was added Sep. 20 1996. The $classes and $ids arguments were added
in version 5.01.1219783000 20080826. There must be at least one $name value.

SEE ALSO

options, checkbox
1.5.46 checkbox – print checkbox list

SYNOPSIS

<checkbox $name $values $selected $output
  [$classes [$ids [$titles]]]>

DESCRIPTION

The checkbox function generates a list of checkboxes (HTML `<INPUT TYPE=checkbox>` tags), one for each value of $output. The $name list is the name of the tag; if it has fewer values than $output, the last name is re-used. The $values list is the list of VALUE attribute values. Values that match a value of $selected are set CHECKED. The $output list is the list of user-visible values. If $values has fewer values than $output, the corresponding $output values are used.

In version 5.01.1219783000 20080826 and later, the output is XHTML 1.0 Strict compliant, and the optional $classes and $ids arguments may be given. The $classes argument is a parallel list of class attribute values to set; if the list is short, the last value is re-used. If no values (or a single empty value) is given, no class attributes are set.

The $ids argument is a parallel list of HTML id attribute values to set; if the list is short, the last value is re-used, but with its last numeric section incremented each time (to make it unique). This allows control of the starting id value, yet still lets the function handle the rest of the sequence. If no values (or a single empty value) is given, the ids are named after $name with an incrementing number.

In version 5.01.1225758000 20081103 and later, an optional $titles argument may be given. If non-empty, each value specifies the title attribute to set for the corresponding item.

Each $output value is printed in a `<label>` associated with its checkbox via the id, so that clicking on the displayed text also toggles the checkbox, making it easier to click.

DIAGNOSTICS

checkbox returns nothing.

EXAMPLE

<$values = red green yellow blue><$selected = green>
<checkbox "color" $values $selected $values>

CAVEATS

The checkbox function was added Sep. 20 1996. The $classes and $ids arguments were added in
version 5.01.1219783000 20080826. There must be at least one $name value.

SEE ALSO

options, radiobutton
1.5.47  doctype – print DOCTYPE declaration

SYNOPSIS

<doctype $langver [$type]>

DESCRIPTION

The <doctype> function prints a <!DOCTYPE> declaration based on the $langver and $type arguments, which are shorthand aliases for some common declarations. Using this function makes code more compact, since the entire standard declaration need not be entered, as well as helps check for errors/typos. The possible values for $langver are:

- html2.0 – For HTML 2.0
- html3.2 – For HTML 3.2
- html4.01 – For HTML 4.01; a $type must be given also
- html5 – For HTML 5
- xhtml1.0 – For XHTML 1.0; a $type must be given also
- xhtml1.1 – For XHTML 1.1

The possible values for $type are:

- strict – For the Strict DTD
- transitional or loose – For the Transitional DTD
- frameset – For the Frameset DTD

DIAGNOSTICS

dofctype returns 1 on success, or 0 on error.

EXAMPLE

<doctype "xhtml1.0" "loose">

CAVEATS

The <doctype> function was add in version 5.01.1220042000 20080829.
1.5. BUILTIN FUNCTIONS

1.5.48 cal – print a calendar with links

SYNOPSIS

<cal [options] [$dates [$events [$var ...]]][ /]> [</cal>]

DESCRIPTION

The cal function prints one or more calendars. It takes an optional list of $dates which are significant dates to note on the calendar. (If no $dates are given, the calendar defaults to the current month.) The $dates may be Texis-parseable dates, counter values, or a vCalendar/English calendar rule (see the SCHEDULE tag for syntax, p. 71). The range of dates also determines the month(s) that the calendar spans, unless otherwise specified (see options). A parallel argument $events is an optional list of short text names for these dates, to be printed in the appropriate day. Further parallel variables can be given ($var etc.), which will be looped over in the same fashion as $dates and $events. These can be used for ancillary data associated with each event.

Each day number will be linked to a function (see the DAYFUNC option), as well as each event (see EVENTFUNC). Each day link will have a variable ($date) set to that day’s date. Each event link will have the appropriate $dates, $events etc. variables’ value looped over. Thus, if the programmer EXPORTs the desired variables to URL, then these links can be used to provide a detailed view of a particular day or event that the user clicks on.

If a close tag (</cal>) is given, the function is looping. In this case only the header, footer and row ends (<tr>/</tr>) are printed for the calendar, and the programmer is responsible for printing each day of the calendar inside the loop (i.e. <td>, text and </td>). This allows more flexibility as to how each day is displayed. Every day of every month of the calendar is iterated over. Inside each iteration, 0 or more of the appropriate values for this day of the $dates, $events, etc. variables are available. (It is important to note that inside the <cal> loop, these variables are not in a loop context yet, because more than one event, or zero events, may occur on a given day. The programmer should <loop> over these variables inside the <cal> loop.) The special variables $loop and $next are set and incremented each <cal> iteration, starting with 0 and 1 respectively. The variable $date is also set to the date of the current day.

Options that can be specified include:

- TYPE=type
  Sets the type of calendar to print. The possible values are week, month, quarter, and year. A month calendar displays one or more contiguous months. A quarter calendar displays one or more contiguous quarters, each three months across and starting with January, April, July or October as appropriate. A year calendar displays one or more contiguous years, starting with January. A week calendar displays a range of days like the month type, but later months are contiguously printed in the same box without starting a new box. The default type is the smallest of month, quarter or year that is needed to display the entire range of $dates (if possible). (Note: Prior to version 6.00.1298435000 20110222, the month, quarter and year types only ever displayed one month, quarter or year, regardless of events or start/end range.)
• **DATEVAR=**date
  Sets the name of the variable to set at each iteration to the current day’s date. The default is “date”, e.g. the variable $date is set each iteration. If the programmer EXPORTs this variable to the URL, it will be available to each link the user clicks on.

• **DAYFUNC=/day.html**
  Sets the function/mime extension to append to the link for each day number in the calendar. The default is “/day.html”, i.e. the function day will be called when a day number is clicked on. Note that this option is irrelevant in the looping syntax, since it is up to the programmer to explicitly generate all text for a day, including links.

• **EVENTFUNC=/event.html**
  Sets the function/mime extension to append to the link for each event in the calendar. The default is “/event.html”. This option is irrelevant in the looping syntax, since the programmer must generate all links.

• **MAXEVENTS=5**
  The maximum number of events to display in a day. The default is 5; if many events fall on one day there may not be enough room to show them all, and too many may distort the calendar. This option is irrelevant to the looping syntax: it is up to the programmer to set a MAX on the LOOP over $events.

• **BORDER=1**
  The value for the border of the <table> tag for the calendar. The default is 1; giving an empty string generates no border attribute.

• **WIDTH="100%"**
  The value for the width of the <table> tag for the calendar. The default is “100%”; giving an empty string generates no width attribute.

• **HEIGHT="50%"**
  The value for the height of the <table> tag for the calendar. The default is “50%” (except for the WEEK type where it is based on the number of weeks printed); giving an empty string generates no height style.

• **CELLPADDING=1**
  The value for the cellpadding of the <table> tag for the calendar. The default is 1; giving an empty string generates no cellpadding attribute.

• **CELLSPACING=0**
  The value for the cellspacing of the <table> tag for the calendar. The default is 0; giving an empty string generates no cellspacing attribute.

• **WEEKLEN=1**
  The maximum length of the weekday-name strings to print. The default is the entire length for week and month types, and 1 for the quarter and year types.

• **MONTHFMT="%B %Y"**
  The date format to print the month header with. The default is “%B %Y”. (Note: Prior to version 6.00.1298435000 20110222, the YEAR type default was “%b”.)
• **DAYFMT=**"%e"
The date format to use for each non-event day number. The default is “%e” for Unix, and “%#d” for Windows.

• **NEWDAYFMT=**"%b %e"
The date format to use for each day number that begins a new month, in the week type. The default is “%b %e” for Unix, and “%b %#d” for Windows.

• **EVENTFMT=**"<b>%e</b>"
The date format to use for each day number that has events. The default is “<b>%e</b>” for Unix, and “<b>%#d</b>” for Windows.

• **MONTHCLASS=**class
The style class attribute for the <td> for the month header. The default is to center it with a larger font.

• **WEEKCLASS=**class
The style class attribute for each <td> for the weekday names. The default is MONTHCLASS if given, otherwise they are printed with <th> cells.

• **DAYCLASS=**class
The style class attribute for each <td> for non-event days. The default is none.

• **EVENTCLASS=**class
The style class attribute for each <td> for event days. The default is DAYCLASS if given, otherwise none.

• **START=**date
The starting date for the calendar. The default is the start of the week, month, quarter or year (depending on TYPE) of the earliest value of $dates. For the week type, day resolution is possible; all other types always start on an integral month, quarter or year. (Note: Prior to version 6.00.1298435000 20110222, the month, quarter and year types only ever displayed one month, quarter or year, regardless of events or start/end range.)

• **END=**date
The ending date for the calendar. The default is the end of the week, month, quarter or year (depending on TYPE) of the latest $dates value. For the week type, day resolution is possible; all other types always end on an integral month, quarter or year. (Note: Prior to version 6.00.1298435000 20110222, the month, quarter and year types only ever displayed one month, quarter or year, regardless of events or start/end range.)

• **NOHREF**
If given, this flag turns off the day and event links. This can be used to save output space if the links are not to be used. It is irrelevant in the looping syntax since there the programmer is responsible for links.

**EXAMPLE**

This example prints a one-month calendar for December with some events noted:
This example uses the looping syntax to print a similar calendar as above, except that days with events have a gray background, and only events are linked:

```xml
<$dates = "Dec 25" "Dec 31" "Dec 7"$>
<$events = "Christmas" "New Year’s Eve" "Pearl Harbor Day"$>
<cal $dates $events>
  <$IF $dates neq ""$>
    <td align="left" valign="top"
        style="height: 15%; width=14%; background-color: gray">
  <$ELSE$>
    <td align="left" valign="top"
        style="height: 15%; width=14%">
  <$ENDIF$>
  <strfmt "%at " "%d" $date$><sandr "$ret$>$ret<br/>
  <$LOOP $dates $events>
    <a href="$url/event.html">$events</a>
  <$LOOP$>
  </td>
</cal>
```

**CAVEATS**

The `cal` function was added in version 3.01.970250000 20000929.

Prior to version 6.00.1298435000 20110222, the `$url` variable must be explicitly used in the script for links to work properly.

The `DATEVAR`-named variable (`$date` by default) must be explicitly used in the script for it to be exported properly.

**SEE ALSO**

`calrule`
1.5. BUILTIN FUNCTIONS

1.5.49 calrule – translate a calendar rule and iterate dates

SYNOPSIS

<calrule [options] [RULE=$rule[ /]> [</calrule>]

DESCRIPTION

The calrule function takes a vCalendar/English calendar rule (see the SCHEDULE directive, p. 71), and returns the Texis date(s) corresponding to the rule. This function can be used to find the actual fixed date(s) a rule occurs on, e.g. for lookup in a fixed-date events table. The dates are returned in $ret.

If an optional end tag is given, the function becomes a looping statement, and statements inside the <calrule> ... </calrule> block are executed for each return value. The special variables $loop and $next are then set as in other looping statements.

Options that can be specified include:

- RULE=$rule
  The vCalendar or English calendar rule to iterate over.

- START=$date
  The Texis-parseable starting date for the rule, if required. (Note that some rules are absolute-dated and may ignore this.) The rule starts on the first applicable date on or after this date. For example, if the rule were “every Sunday at 6pm”, then a START date of “2001-01-01” would make the rule return dates only on or after Jan. 1, 2001. Defaults to “now”. Ignored if TOVCAL or TOENGLISH set. Note: this is different from the FIRST date, which is the starting date for iteration and has nothing to do with the calendar rule.

- END=$date
  The Texis-parseable ending date for the rule, if required. Default is infinite. Ignored if TOVCAL or TOENGLISH set. Note: this is different from the LAST date, which is the ending date for iteration and has nothing to do with the calendar rule.

- SKIP=$n
  Number of initially returned result dates to skip. Default is 0.

- MAX=$n
  Max number of results to return. The default is 100, since some rules can produce many result dates (e.g. “every 10 minutes”).

- FIRST=$date
  The Texis-parseable date to start returning results after. In other words, skip any dates that would be returned before this date. Defaults to negative infinity (e.g. Jan. 1 1970). Note that this is different from the START date, which is associated with the calendar rule itself.
• LAST=$date
  The Texis-parseable date to stop returning results before. In other words, this is the maximum date to return results on or before. Defaults to infinite. Note that this is different from the END date, which is associated with the calendar rule itself.

• ROW
  As with other looping statements, do not accumulate results in $ret: each new value overwrites the previous, and $ret is not a loop variable.

• TOVCAL
  Instead of returning dates, translate the English $rule to vCalendar syntax and return it. Mutually exclusive with the TOENGLISH flag.

• TOENGLISH
  Instead of returning dates, translate the vCalendar $rule to pseudo-English syntax and return it. Mutually exclusive with the TOVCAL flag. Note that the returned syntax may not be entirely reverse-compatible.

EXAMPLE

<$rule = "every Sunday at 6pm">
The next 10 dates the event "$rule" occurs are:
<calrule ROW MAX=10 RULE=$rule>
  Occurs on: $ret
</calrule>

CAVEATS

The calrule function was added in version 4.0.997700000 20010813.

SEE ALSO

cal, SCHEDULE
1.5. BUILTIN FUNCTIONS

1.5.50 calendar – print calendar form

SYNOPSIS

<calendar $yr $month $day $hour $min ["am"|"pm" [$prefix [flags ...]]]>
<calendar $date [$prefix [flags ...]]>

DESCRIPTION

Note: The cal function (p. 251) is more flexible and has more options than the calendar function.

The calendar function prints a calendar for the given month, as an HTML form that can be used to select a date and time.

In the first form, the year, month, day, hour and minute are given as integers. The months start from 1, and the hour is assumed to be 24-hour unless “am” or “pm” is given after the minute. In the second form, the date is given as a single field. It can be a Texis date field, or any string recognized by Texis as a date such as “now” or “-5 weeks”.

The calendar is printed in an HTML TABLE, with form fields calyear, calmonth, calday, calhour and calmin. The calday field is a radio button for each day of the month. The variable name prefix can be changed by giving the $prefix parameter, which is the string to use instead of cal. This can be used to distinguish fields if multiple calendars are given in one form.

The following flags are recognized and can be given after the prefix:

- “checkbox”
  Use checkboxes instead of radio buttons for the days.

- “radio”
  Use radio buttons for the days (default).

- “julian”
  Print Julian days (days into the year) instead of days of the month.

Upon submission of the form, the caldate function can be called to construct a list of Texis date values from the submitted values.

DIAGNOSTICS

calendar returns nothing.

EXAMPLE
<FORM METHOD=post ACTION=$url/action.html>
  Choose a day this month:
  <calendar "now">
  <INPUT TYPE=submit>
</FORM>

CAVEATS

The calendar function was added Oct. 4 1996.

SEE ALSO

caldate
1.5. BUILTIN FUNCTIONS

1.5.51 caldate – create date list from calendar submission

SYNOPSIS

<caldate $varprefix>

DESCRIPTION

The caldate function returns a list of dates from the variables submitted by a form created with calendar. The value(s) of $varprefix are the variable prefix(es) given to calendar.

EXAMPLE

This script generates a calendar, and lists the date(s) checked off upon submission:

<A NAME=main>
  <FORM METHOD=post ACTION=$url/list.html>
    <calendar "now" "cal" "checkbox">
      <INPUT TYPE=submit>
    </FORM>
  </A>

<A NAME=list>
  <caldate "cal">
    You submitted:
    <LOOP $ret> $ret <BR> </LOOP>
  </A>

CAVEATS

The caldate function was added Oct. 4 1996.

SEE ALSO

calendar
1.5.52 clist – create comma-separated list

SYNOPSIS

<clist $values>

DESCRIPTION

The `clist` function returns the given list of values as one parenthetical, comma-separated value. If the list is empty, the returned value is empty too, instead of `()`. This function can be used to create a Metamorph comma-separated values list. Note that a multi-value string variable is automagically turned into comma-separated values when used as an argument in a SQL command (p. 27).

DIAGNOSTICS

clist returns the $values in one parenthetical, comma-separated value, or an empty string if there are no values.

EXAMPLE

```text
<$list = "red" "blue" "violet">
<clist $list>
<fmt "%mIH" $ret "Roses are red.">

$ret would have the value "(red,blue,violet)" after clist is called, and the output would be "Roses are <B>red</B>.".
```

CAVEATS

The `clist` function was added Sep. 10 1996.

Note that a comma-separated list is a single set element in Metamorph. See the Metamorph documentation for details. Only parentheses are escaped in the output. Thus, if a value has a comma in it, it will be treated as multiple values in the list.

SEE ALSO

slist
1.5.53  slist – create Metamorph set list

SYNOPSIS

<slist $intersect $values>

DESCRIPTION

The slist function returns the given list of values as one Metamorph query with $intersect intersections. Each value is double-quoted. If the list is empty, the returned list is empty too.

This function can be used to create a Metamorph query for markup using fmt (p. 92).

DIAGNOSTICS

slist returns a Metamorph query for $intersect intersections of the values of $values.

EXAMPLE

<$list = "red" "blue" "violet">
<slist 1 $list>
<fmt "%mIH" $ret "Roses are red, not blue."/>

The output would be:

Roses are <B>red</B>, not <B>blue</B>.

The query executed ($ret) was @1 "red" "blue" "violet".

CAVEATS

The slist function was added Sep. 10 1996.

The values of $values should not contain non-ASCII characters.

SEE ALSO

clist, Metamorph documentation
1.5.54 wordlist, wordcount, wordoccurrencecounts – get words and frequencies from index

SYNOPSIS

<wordlist $table [$field [$wordsOrWildcards [$options]]]>
<wordcount>
<wordoccurrencecounts>

DESCRIPTION

The wordlist function returns a list of the words in the given $table, as found in a Metamorph index. The first Metamorph index found is used, for $field if given, otherwise any field. Each value of $wordsOrWildcards can be a single word, in which case only that word is returned, or a word-prefix followed by “*” (asterisk), in which case only words having that prefix are returned. If $wordsOrWildcards is not given, all words are returned.

The following values for $options are accepted:

- NOCOUNTS
  No counts are available via wordcount. This is to conserve memory when examining a large list when the counts are not needed.

- db $db
  Look for $table in database $db, instead of the current database. Note that these must be consecutive values of a single $options argument. Added in version 7.05.1459800000 20160404.

The wordcount function returns a list of the row counts of each corresponding word returned by the previous wordlist, i.e. the number of rows each word occurs in.

In version 6 and later, the wordoccurrencecounts function returns a list of the occurrence counts of the corresponding words. E.g. if a word occurs twice in each of 10 documents, its wordcount value will be 10, while its wordoccurrencecounts value will be 20. Note that word occurrence information is only stored for inverted Metamorph indexes: non-inverted indexes will return 0 or nothing for word occurrence values.

DIAGNOSTICS

wordlist returns a list of the words found in a Metamorph index. wordcount returns the corresponding document frequencies of those words. wordoccurrencecounts returns the hit counts (every word every doc).

EXAMPLE
This example prints a list of the words and their frequencies in the title field of the table books, sorted by ascending frequency (e.g. rarest first):

```
<wordlist "books" "title">$words = $ret
<wordcount>
<sort $ret $words>
<LOOP $ret $words>
  $words $ret
</LOOP>
```

CAVEATS

The wordlist and wordcount functions were added Feb. 20 1997. The NOCOUNTS option was added in March 1999.

A Metamorph index must exist on the named table/field for wordlist to work. Note that what constitutes a word, and how many words there are, is dependent on the Metamorph index, how it was created (e.g. the index expression), and when it was last updated.
CHAPTER 1. TEXIS WEB SCRIPT

1.5.55  createdb – create database

SYNOPSIS

<createdb $db $syspass $publicpass>

DESCRIPTION

The createdb function creates a new database whose path is given by $db. The passwords for _SYSTEM and PUBLIC are given as arguments. For security, the _SYSTEM password $syspass is required to be at least 6 characters and contain punctuation. The PUBLIC password $publicpass may be empty.

Note that unlike the DB directive, which affects all other SQL commands, the database for createdb is specified at run time. Thus, createdb can be used in an administrative script to create new databases.

DIAGNOSTICS

createdb returns “Creating database succeeded” on success, or “Creating database failed” or nothing on error.

EXAMPLE

<createdb "/usr/local/mydb" "Klaatu!" "">

CAVEATS

The createdb function was added Oct. 10 1996.

The path given to createdb is absolute, not relative to the HTML documents tree, and is not checked. It is up to the programmer to verify the path before calling createdb, e.g. to check whether the Web user is permitted to access the dir, etc.

SEE ALSO

adminsql
1.5. BUILTIN FUNCTIONS

1.5.56  adminsql – execute arbitrary SQL

SYNOPSIS

<adminsql [options ...] [statement ...]>

DESCRIPTION

The adminsql function executes an arbitrary SQL command, constructed by concatenating its arguments together. The result rows of the command are printed in an HTML table, with the columns named after the fields. Unlike the SQL statement, there is no restriction on returned field names being valid Vortex variable names (e.g. convert(id, 'date') is ok). However, the fields are not returned; they are simply printed out. The arguments to adminsql are appended together as one string and executed. Note that this removes the argument protection afforded by the SQL statement. Any SQL parameters, like field values, are given as literals, e.g. string values must be in single quotes.

Several options are available. Note that since <adminsql> is implemented as a user (not builtin) function, options that take arguments must be quoted atomically with no space, e.g. "MAX=5".

  • SKIP=N
    Skips the first N result rows (does not print them). Added in version 5.01.1197080000 20071207.

  • MAX=N
    Prints at most N result rows (not including SKIP).

DIAGNOSTICS

adminsql returns nothing.

EXAMPLE

<$tbl = "SYSTABLES">  <adminsql "select" " * from " $tbl>

CAVEATS

The adminsql function was added Oct. 10 1996.

Note: No checking of the SQL is done before execution; it is the caller’s responsibility. The caller must ensure that the statement is correct, and no “rogue” SQL is being inserted (aka “SQL injection”), e.g. a value of “; DROP TABLE xyz” for some field value. The parameter protections of the SQL statement (p. 27) are not applied to adminsql’s arguments. It is safer to use the SQL statement where possible.
which can protect against SQL injection; `adminsql` is primarily useful only for tables with unknown column names (e.g. “SELECT *”).

SEE ALSO

SQL, `createdb`
1.5. BUILTIN FUNCTIONS

1.5.57 loguser – log user accesses

SYNOPSIS

$loguser $REMOTE_ADDR [$limit]$

DESCRIPTION

The loguser function adds 1 to the count of accesses in the last minute by the host $REMOTE_ADDR,
which is a dotted-decimal IP address unique to the current user. The web server sets the $REMOTE_ADDR
environment variable to the address of the remote host.

In combination with the userstats function, loguser provides a way to limit access by overzealous or
rogue users. The access count for users is shared across all Vortex scripts. Any script can update the access
count for a user with loguser, and all scripts will be instantly aware of the count.

DIAGNOSTICS

loguser returns “Success”, or “Fail” if more than $limit (default 5) distinct users have accessed
the server in the past minute (as counted by loguser). The count for the given user is updated in either
case.

EXAMPLE

$loguser $REMOTE_ADDR$

CAVEATS

The loguser function was added Oct. 10 1996. In version 2.1.882170000 19971215 and later the first
parameter is required to be an IP address, and the second argument is accepted.

SEE ALSO

userstats, resetstats
1.5.58  userstats – get user statistics

SYNOPSIS

```
<userstats $REMOTE_ADDR [license|user]>
```

DESCRIPTION

Returns access statistics for the user $REMOTE_ADDR, a dotted-decimal IP address.

The first argument is normally $REMOTE_ADDR, i.e. the current remote host. In this case, the average number of accesses per minute by that user is returned; this count is reset if the user is idle more than a minute. If the first argument is the empty string, then the number of users in the last minute is returned. If the first argument is “high”, then the highest number of users ever seen, since initialization (machine boot), is returned.

The optional second argument determines which set of statistics are queried: “user” (the default) queries hits logged with loguser. “license” queries the license statistics, i.e. every Vortex usage.

DIAGNOSTICS

userstats returns the (integer) number of accesses by the given user in the last minute, or the current or highest number of active users. license gives builtin stats, user gives stats logged by <loguser>.

EXAMPLE

```
<loguser $REMOTE_ADDR>
<userstats $REMOTE_ADDR>
<IF $ret gt 20>
    You’re accessing this server too fast. Please slow down. <BR>
    <flush><sleep 10>       <!-- slow ’em down -->
</IF>
```

CAVEATS

The userstats function was added Oct. 10 1996. In version 2.1.882170000 19971215 and later the first parameter is required to be an IP address, and the second argument is accepted.

SEE ALSO

loguser, resetstats
1.5.59  resetstats – reset user statistics

SYNOPSIS

<resetstats $REMOTE_ADDR>

DESCRIPTION

The resetstats function resets the user access count – maintained by loguser – for the given user to 0. If the $REMOTE_ADDR argument is empty, all users are reset.

DIAGNOSTICS

resetstats returns nothing.

EXAMPLE

<!-- reset stats for localhost: -->
<IF $REMOTE_ADDR eq 127.0.0.1>
  <resetstats $REMOTE_ADDR>
</IF>

CAVEATS

The resetstats function was added Oct. 10 1996. In version 2.1.882170000 19971215 and later the first parameter is required to be an IP address.
License statistics cannot be reset.

SEE ALSO

loguser, userstats
1.5.60  abstract – generate an abstract from text

SYNOPSIS

<abstract $text [$maxsz [$style [$query]]]>  

DESCRIPTION

The abstract function generates an abstract of a given portion of text. The abstract will be less than $maxsz characters long, and will attempt to end at a word boundary. If $maxsz is not specified (or is less than or equal to 0) then a default size of 230 characters is used. The $style argument allows a choice between several different ways of creating the abstract. Note that some of these styles require the $query argument as well, which is a Metamorph query to look for:

- dumb (0)
  Start the abstract at the top of the document.

- smart (1)
  This style will look for the first meaningful chunk of text, skipping over any headers at the top of the text. This is the default if neither $style nor $query is given.

- querysingle(2)
  Center the abstract contiguously on the best occurrence of $query in the document.

- querymultiple(3)
  Like querysingle, but also break up the abstract into multiple sections (separated with “…” ) if needed to help ensure all terms are visible. Also take care with URLs to try to show the start and end.

- querybest
  An alias for the best available query-based style; currently the same as querymultiple. Using querybest in a script ensures that if improved styles become available in future Vortex releases, the script will automatically “upgrade” to the best style.

If no $query is given with a query... mode, it falls back to dumb mode. If a $query is given with a non-query... mode (dumb/smart), the mode is promoted to querybest. The current locale and index expressions also have an effect on the abstract in the query... modes, so that it more closely reflects an index-obtained hit.

If there are fewer values of $maxsz or $style than $text, the last value(s) are re-used.

DIAGNOSTICS

abstract returns an abstract for each of the supplied texts.
EXAMPLE

<LOOP $docid $fulltext>
  <A HREF=$url/fulldoc.html>View full-text</A> <P>
  <abstract $fulltext 230 querybest $query>
  $ret
  <HR>
</LOOP>

CAVEATS

The abstract function was added Oct. 24 1996.

The querymultiple and querybest styles were added in Version 6.
1.5.61  rmcommon – remove common prefix/suffix from text

SYNOPSIS

<rmcommon $data $template [$maxrm]>

DESCRIPTION

The rmcommon function removes the prefix and suffix text from each $data value that is shared with the corresponding $template value. Up to $maxrm characters are removed, rounded down to the nearest word boundary; the default is the maximum amount of common text. This function is useful in stripping common header and footer text from web pages before indexing.

DIAGNOSTICS

rmcommon returns $data with its common prefix/suffix text removed.

EXAMPLE

<$template = "Acme Industries, Inc. [Data] Home Next Previous">
<rmcommon $data $template>
<SQL NOVARS "insert into webpages
values(counter, $Url, $ret)">
</SQL>

In the above example, $template is set to a template representative of a typical (formatted) page from a web site, i.e. an actual fetched page. Like all pages from this site, it contains the same title prefix and navigation-bar suffix that we want to strip before indexing, to prevent useless hits on “Acme” for example. By using this template with <rmcommon> against every fetched page $data, the prefix/suffix is stripped before insertion into the database. Thus, if $data was initially “Acme Industries, Inc. Widgets and Gadgets Home Next Previous”, after the <rmcommon> call it would be inserted as “Widgets and Gadgets”.

CAVEATS

The rmcommon function was added in version 3.01.984600000 20010314.
1.5. BUILTIN FUNCTIONS

1.5.62 pwencrypt – encrypt Unix-style password

SYNOPSIS

<pwencrypt $pass $salt>

DESCRIPTION

The `pwencrypt` function generates an encrypted Unix-style password for each password value in `$pass`. The corresponding value of `$salt` is used as the two-character salt; if it is empty ("") a random salt is generated. There must be one or more values of `$pass` and `$salt`; if fewer values of `$salt` exist than `$pass`, the last `$salt` value is re-used. Only the first 8 characters of `$pass` and the first 2 characters of `$salt` are used.

DIAGNOSTICS

pwencrypt returns each `$pass` password, encrypted Unix-style. A random salt is used if `$salt` is empty.

EXAMPLE

<!-- $enpass is Unix /etc/passwd encrypted password field; -->
<!-- $pass is user password -->
<pwencrypt $pass $enpass>
<IF $ret eq $enpass>
  Login ok.
<ELSE>
  Bad login.
</IF>

CAVEATS

The `pwencrypt` function was added Oct. 25 1996.

SEE ALSO

encrypt, decrypt
1.5.63 encrypt – encrypt data

SYNOPSIS

<encrypt $data $key>

DESCRIPTION

The encrypt function encrypts each value of $data using the corresponding $key value. There must be one or more values of $data and $key; if fewer values of $key exist than $data, the last $key value is re-used.

DIAGNOSTICS

encrypt returns each value of $data, encrypted Unix style, using key $key.

EXAMPLE

<encrypt $data $pass>
<SQL "insert into secret values(counter, $ret)" /></SQL>

CAVEATS

The encrypt function was added Oct. 25 1996.

The encryption algorithm uses only the first 8 bytes of each key. The returned encrypted data is likely to be binary, but will not contain nul (ASCII zero) values.

SEE ALSO

decrypt,pwencrypt
1.5.64 decrypt – decrypt data

SYNOPSIS

<decrypt $crypt $key>

DESCRIPTION

The decrypt function decrypts each value of $crypt (previously encrypted with encrypt) using the corresponding $key value. There must be one or more values of $crypt and $key; if fewer values of $key exist than $crypt, the last $key value is re-used.

DIAGNOSTICS

decrypt returns each value of Unix-style encrypted data $crypt, decrypted using key $key.

EXAMPLE

<SQL "select data from secret where id = $id"> </SQL>
<decrypt $data $pass>
The data is: $ret

CAVEATS

The decrypt function was added Oct. 25 1996.

The encryption algorithm uses only the first 8 bytes of each key.

SEE ALSO

encrypt, pwencrypt
1.5.65  readvars – read URL-encoded variables

SYNOPSIS

<readvars $srcvarnames [$destvarnames] $urldata>

DESCRIPTION

Vortex variables are automatically imported from the CGI query string (QUERY_STRING environment variable) on script start, so normally no special parsing is needed. However, in cases where different or custom URL-encoded data must be parsed for values, the <readvars> function can be used. readvars assigns Vortex variables from the URL-encoded string(s) in $urldata.

If the $srcvarnames parameter is non-empty, then only those $urldata variables named as values of $srcvarnames will be read; the default ($srcvarnames empty) is to assign all variables in $urldata. This prevents a script’s reserved variables from being overwritten by a rogue or unknown $urldata string.

In version 6.00.1283296000 20100831 and later, an optional $destvarnames argument may be given. This is a list of Vortex variable names to assign the parallel $srcvarnames query string variables to. It provides a way to parse an arbitrary or changing list of query string variable names into a fixed set of known Vortex variables.

DIAGNOSTICS

The readvars function returns the number of values that were URL-decoded and assigned from $urldata. $srcvarnames is the list of valid $urldata variable names; $destvarnames is the optional list of Vortex variables to assign to.

EXAMPLE

<<str = "Don’t change me">
<<data = "x=123&str=hello+there&x=5"> <!-- e.g. from user -->
<readvars "x" $data>

$x would have the values 123 and 5, but $str would still be “Don’t change me” because “str” was not one of the variable names specified to readvars.

CAVEATS

The readvars function was added Nov. 14 1996. Full binary data is not yet supported. If a variable named in the $urldata parameter is unknown in the script (i.e. is not explicitly referenced elsewhere), it
will not be assigned since it doesn’t exist. There is as yet no equivalent \texttt{writevars} function; use \texttt{fmt} (p. 92) with the \texttt{\%U} format.

\textbf{SEE ALSO}

\texttt{fmt \"\%U\"}, \texttt{fmt \"\%!U\"}, \texttt{getvar}, \texttt{setvar}
1.5.66  varinfo – get miscellaneous variable information

SYNOPSIS

<varinfo $action [flags ...] [$arg]>

DESCRIPTION

The varinfo function returns various information about variables, depending on the value of the $action argument:

- **list**
  Returns the names of all known variables in the current scope. This includes unlisted environment/CGI variables only accessible via getvar. If the optional $arg argument is given, then only variables from a specific source will be returned:

  - **URL** for URL (query string) variables
  - **ENV** for environment variables
  - **CONTENT** for POST method form variables
  - **CGI** for POST and/or query-string variables
  - **COOKIE** for cookie variables

  Each of the returned variable names can be passed to getvar to obtain its value(s).

- **size**
  Returns the size in bytes of each corresponding value of the variable named by $arg. This is useful to obtain the true size of a varbyte variable such as an image, where strlen might stop short.

- **type**
  Returns the SQL type of the variable named by $arg, as a string. The type action was added in version 2.6.929000000 19990610.

- **dump**
  Returns debug information about the variable named by $arg, as a string. This information is in the same format as printed by the $?myVar debug syntax (p. 7). Accessing debug info via <varinfo dump> can be used when the variable name must be determined dynamically at run-time, precluding usage of the easier (but static-name) $?myVar syntax. Added in version 6.

- **dumpverbose**
  Like dump, but more verbose: uses the $??myVar syntax. Added in version 6. In version 7.03 and later, this is deprecated: use <varinfo dump verbose ...> instead.

The following actions are applicable only to <INPUT TYPE=file> variables from a multipart MIME file upload (p. 6). For any other variables, nothing is returned:
1.5. BUILTIN FUNCTIONS

- **filename**
  Returns the filename for the variable named by $arg, from the Content-Disposition header. This corresponds to the name of the file the user gave for that field, from their system. Note that the full path is generally not available; only the filename is usually sent by the browser.

- **contenttype**
  Returns the Content-Type for the variable given by $arg.

In version 7.03 and later, the following flags may be set before the final $arg argument. Currently they are only valid for the dump and dumpverbose actions:

- **esc**
  Force HTML-escapement of variable values printed by dump, regardless of HTML mode.

- **noesc**
  Force as-is (no HTML escapement) printing of variable values printed by dump. By default, HTML escapement happens if the script is in HTML mode; using this flag may be useful to avoid escapement when an HTML-mode Web script is being debugged to a log file that is being read as plain text.

- **verbose**
  Dump variables in verbose mode (ala $??myVar).

**DIAGNOSTICS**

The varinfo function returns various information about variables, per its arguments.

**EXAMPLE**

```html
"A NAME=main"
  <FORM METHOD=post ACTION="$url/test.html"
    ENCTYPE="multipart/form-data">
    Image: <INPUT TYPE=file NAME=image>
    <INPUT TYPE=submit NAME=submit>
  </FORM>
"A"

"A NAME=test"
  <varinfo list CGI>
  Form variables are: <LOOP $ret> $ret </LOOP> <BR>
  <varinfo filename "image">
  You uploaded the file "$ret".
  <varinfo contenttype "image">
  It is of type "$ret" and
  <varinfo size "image"> $ret bytes in size.
  </varinfo size "image">
"A"
```
CAVEATS

The \texttt{varinfo} function was added in version 2.6.926900000 19990517.

The size of variables is the size of the native type, not the string representation as \texttt{strlen} would return. E.g. integer vars on a 32-bit machine will always have a size of 4.

While the \texttt{list cgi} action will return CGI variable names, their values, obtained with the \texttt{getvar} function, may be different if the variable(s) have been modified. Similarly, the \texttt{filename} and \texttt{contenttype} of variables may not correspond to their current values, if they were modified.

SEE ALSO

\texttt{getvar}
1.5.67 getvar – get variable values

SYNOPSIS

<getvar [NOMSG] $var>

DESCRIPTION

The getvar function returns the values of the (single) variable named by getvar. It can be used in the rare situations where the name of a variable cannot be determined until run time, i.e. $var syntax cannot be used because the var name dynamically changes.

If NOMSG is given, it will not issue a putmsg if the given name is invalid and not found. This flag was added in Texis version 7.

DIAGNOSTICS

The getvar function returns the values of the (single) named variable.

EXAMPLE

<A NAME=main>
  <FORM METHOD=post ACTION="$url/test.html">
    First name: <INPUT NAME=First VALUE="$First">
    Last name: <INPUT NAME=Last VALUE="$Last">
    ... City, State, etc. ...
    <INPUT TYPE=submit>
  </FORM>
</A>

<A NAME=test>
  <$vars = $ret>
  <LOOP $vars>
    <getvar $vars>
      <rex "[^\alnum\space]+" $ret>
      <IF $ret neq "">
        Illegal character in $vars field.
      </IF>
    </LOOP>
  </A>
In this example, a form with many text fields is submitted. Each field must be checked for illegal (non-alphanumeric) characters. Rather than laboriously call a function multiple times with hard-coded references to $First, $Last, $Street, etc., the list of form variables is dynamically obtained with varinfo, and each form variable is checked with getvar.

CAVEATS
The getvar function was added May 7 1997.

Where possible, it is more efficient to refer to variables directly by name than via getvar.

Non-CGI variables that are not explicitly used in the script somewhere with a $-sign, e.g. dynamically made-up variable names, cannot be used.

SEE ALSO
setvar, readvars, varinfo
1.5.68 setvar – set a named variable

SYNOPSIS

<setvar [flags] $var $values>

DESCRIPTION

The setvar function sets the (single) variable named by $var to $values. It can be used in the rare situations where the name of a variable cannot be determined until run time.

Optional flags:

- APPEND - values are appended to the end of the variable’s current value list (and are cast to that type)
- NOMSG - no putmsg will be issued if the value cannot be set

DIAGNOSTICS

The setvar function sets the single variable named by $var to $values, and returns nothing.

EXAMPLE

<SQL MAX=1 "select Name, Value from test where id=$id">
  <setvar $Name $Value>
</SQL>

CAVEATS

The setvar function was added May 7 1997.

If a variable named in a setvar call is not directly used in the script (i.e. is not explicitly referenced with $ elsewhere), it will not have any values since it is unknown.

SEE ALSO

getvar, readvars
1.5.69  vxcp – set Vortex control parameters

SYNOPSIS

<vxcp setting [args ...]>

DESCRIPTION

The vxcp function allows control over miscellaneous Vortex settings at run time. Most of these settings will not need to be altered by typical Vortex scripts. The setting can be one of the following:

- **putmsg**
  Controls what action(s) to take when <putmsg>-capturable messages are generated. Takes two arguments: one of call, log, print, or all (meaning all actions), and a boolean value. The first argument names an action, and the second argument turns the action on or off.

  When a message is generated, it can be printed to the output (print), logged to the vortex.log file (log), or the <putmsg> Vortex function (if defined) can be called (call). The default actions are to log and print the message. However, if a <putmsg> Vortex function is defined, logging and printing are turned off, and the function is called. It is assumed <putmsg> will completely handle the error (including logging somewhere) as the script author desires. (See p. 581 for more on capturing errors.)

  In many scripts, however, it is desirable to not only monitor errors with <putmsg>, but to still log them normally for later perusal. Since <putmsg> turns off logging, it has to be done “manually” in <putmsg> by looping over the messages.

  With <vxcp putmsg>, logging can be turned back on without such a manual loop. Message disposition can also be changed at run-time, e.g. to log certain function’s messages but ignore others’.

  In Texis version 7 and later, the log and print actions may have a third state set: exceptiononly. When exceptiononly is set, the action is enabled only for Vortex putmsg exceptions – situations where the <putmsg> script function cannot be called, e.g. an ABEND. For all other (normal) situations the action is disabled. This allows logging/printing to be turned off normally – so a <putmsg> function can handle it – yet enabled for the abnormal situations where the messages would otherwise be lost. The exceptiononly value cannot be set for the call action.

  The return value from the putmsg setting is the previous setting value(s) (2, 1 or 0 for on, exceptiononly or off) for the named action. This enables a function to change an action temporarily, and restore the old settings without side-effects to the rest of the script. (Note that the setting all returns 3 values, for call, log, and print, in that order.)

  See also the <PUTMSG> compiler directive (p. 65); <vxcp putmsg> overrides it (in most cases).

- **putmsgbuffersize**
  Takes an integer or integer-with-size-suffix argument (e.g. “100KB”) that sets the Vortex <putmsg> buffer size in bytes; the default is 16KB. This buffer is used to hold error messages generated during a Vortex statement until the end of the statement, when the messages can be transferred to Vortex variables and the script’s <putmsg> function called. A statement that generates many messages (e.g.
from tracing) can exceed the buffer size, causing a “Too many messages for putmsg buffer” error and loss of messages; increasing the buffer size may avert this. Returns 1 on success, 0 on error. Added in version 6.00.1332547000 20120323.

- putmsgflags $flags [{$flags ...}] $onOff
  Sets given $flags value(s) on or off. Flags are:
  
  - threadid Show thread ID in square brackets in logged and printed putmsgs.
  - printtime Print timestamp when printing putmsgs; normally only shown when logging.

  The $onOff value is a parallel array of boolean values to set the corresponding flags to; if fewer values given than flags, last value is re-used. Returns previous value(s) of given flags. putmsgflags was added in version 7.06.1479841000 20161122.

- stack
  Takes an integer argument defining the maximum Vortex stack depth; -1 for infinite (no limit). The default is 250.

  It is rare that a script would need to increase its stack depth. An unexpected “Stack overflow” error message usually indicates that a function was infinitely recursive, i.e. kept calling itself (directly or through other functions) without returning. Setting a large or infinite stack limit will cause such a script to rapidly consume memory and CPU, possibly taking the machine down with it.

  The return value is the previous stack limit. See also the <STACK> directive (p. 75); <vxcp stack> overrides it.

- timeout
  Takes an integer (-1 for infinite) indicating the script timeout in seconds from now. Or a Texis-parseable date can be given to indicate a deadline, such as “+1 minute”. Returns the previous timeout, as a date deadline, or -1 if the previous timeout was infinite. See also the <TIMEOUT> directive (p. 59); <vxcp timeout> overrides it. Allows the script timeout to be variable at run-time or on a function-by-function basis.

- timeouttext $text [{$hdrNames $hdrValues}]
  Sets the text/HTML to print at Vortex-script timeout to $text, overriding the text set in the <TIMEOUT> directive (p. 59). Can be used to alter the text for different parts of a script, e.g. HTML vs. XML vs. text output sections. Added in version 7. Returns 1 on success, 0 on error.

  In version 7.06.1541109000 20181101 and later, an optional $hdrNames $hdrValues pair of arguments may be given: these are parallel header names and values to print during the timeout as well. Like the text, these will take effect only on timeout. Additionally, these headers will only be printed if no content has been printed yet at the time of timeout. These can be used to set a different content type and/or status for a timeout vs. the main script output. If no $hdrNames $hdrValues arguments are given, no additional/different headers will be printed on timeout.

- trap
  Debug setting. Takes a boolean or integer argument indicating whether to trap signals or not. Debugging/tech-support use. Returns the previous setting. Overrides the <TRAP> directive; see (p. 79) for flag details.
• **connreset**
  Debug setting. Takes a boolean or integer argument indicating how to trap connection-reset on stdout (i.e. when the remote browser user hits the stop button early). Returns the previous setting. Overrides the `<TRAP>` directive; see (p. 79) for flag details.

• **tracealarm**
  Debug setting. Takes an integer value indicating what tracing messages to issue for alarms; same bit flags as the `-tracealarm` command-line option (p. 572). Flags may change or be added to in future releases. Returns previous setting. Added in version 7.06.1477338000 20161024.

• **tracelib**
  Debug setting. Takes an integer value indicating what tracing messages to issue for shared library loading. Value is a bitwise OR of any of the following values; the default is 0 (i.e. no tracing):
  
  - 0x01: `libpath` expansion
  - 0x02: Library file search and loading
  - 0x04: Symbol/function lookup

  Values may change or be added to in future releases. Returns previous setting. Added in version 7.00.1372360000 20130627.

• **tracepipe**
  Debug setting. Takes an integer value indicating what tracing messages to issue for pipes. Value is a bitwise OR of any of the following values; the default is 0 (i.e. no tracing):
  
  - 0x00000001: After open(), close(), TXcreatethread(), LogonUser() etc.
  - 0x00000002: After select(), WaitForMultipleObjects()
  - 0x00000004: After reads
  - 0x00000008: After writes
  - 0x00000010: After SetEvent(), ResetEvent()
  - 0x00000020: After thread run (at soft exit in thread)
  - 0x00000040: The data read
  - 0x00000080: The data written
  - 0x00010000: Before open(), close(), TXcreatethread(), LogonUser() etc.
  - 0x00020000: Before select(), WaitForMultipleObjects()
  - 0x00040000: Before reads
  - 0x00080000: Before writes
  - 0x00100000: Before SetEvent(), ResetEvent()
  - 0x00200000: Before thread run (at start in thread)
  - 0x00400000: The data buffer before reads
  - 0x00800000: The data buffer before writes

  Values may change or be added to in future releases. Returns previous setting.
1.5. BUILTIN FUNCTIONS

- **htmlmode**
  Takes a boolean argument indicating whether to run in HTML mode or not. This controls whether variables are HTML-escaped when printed as-is, and is on by default if there is no URL file extension (p. 8) or if an HTML or an HTML-like MIME type is set for output, either explicitly via header or implicitly via URL extension. In version 7.01.1390866000 20140127 and later, the MIME types text/xml, application/xml and application/xhtml+xml are also recognized HTML-like MIME types; previous versions only recognized text/html and text/vnd.wap.wml. Returns the previous setting. Added in version 3.01.9824000000 20010217.

- **filemsg**
  Takes a boolean argument indicating whether or not to issue a putmsg for sysutil and some syscp file-oriented actions’ errors. The default is on. This setting can be turned off if a large number of sysutil calls are to be made and errors are expected but benign; this avoids the putmsg for just these functions without the hassle of a Vortex <putmsg> function. Returns the previous setting. Added in version 3.01.9762000000 20001207.

- **execmsg**
  Takes a boolean argument indicating whether or not to issue a putmsg for <EXEC> errors. The default is on. Added in version 5. Note that in version 5 and later errors are now returned in $ret.err, regardless of this setting. Thus if error-checking is being done after <EXEC> calls, execmsg can be turned off to prevent spurious messages, and $ret.err checked for errors.

- **execendiotimeout N**
  Sets the timeout (in seconds) for terminating <exec> I/O threads under Windows. The default is 10 seconds. Returns previous value. Added in version 5.01.1153260000 20060718.

- **libpath**
  Takes a single string argument, which is a colon-separated (Unix) or semicolon-separated (Windows) list of directories to search for loadable modules. Overrides the texis.ini file setting [Texis] Lib Path. Loadable modules currently include the JavaScript and SSL (HTTPS) plugins. The path may include the following keywords that have special meanings:

  - `%INSTALLDIR%` The Texis install directory
  - `%BINDIR%` The Texis binary (executables) directory
  - `%EXEDIR%` The directory of the running executable (or the Texis binary dir, if the former cannot be determined)
  - `%SYSLIBPATH%` The system-dependent dynamic library search path (e.g. LD_LIBRARY_PATH)
  - `% A percent sign

  The default value (if libpath and the texis.ini setting [Texis] Lib Path are unset) is %EXEDIR%:%BINDIR%:%SYSLIBPATH% for Unix or %EXEDIR%:%BINDIR%:%SYSLIBPATH% for Windows. %EXEDIR% was added in version 5.01.1214185000 20080622.

  In version 4.04.1073104616 20040102 and earlier, the above keywords were not supported. Instead, the entire path could be set to the single keyword bin to indicate the installed binary directory (default), or sys to indicate a system-dependent search path. Returns the previous setting.
The `libpath` setting was added in version 4.01.1023500000 20020607. See also the `--lib-path` option (p. 571).

- **libcheckversion $module on|off**
  Controls whether to check version of loadable $module when it is loaded. Loading the wrong version can cause aberrant behavior or an ABEND, so a version check is normally done if possible. This setting can be used to override the check, if a different version of the loadable module is being used that is known to be binary-compatible. The possible values for $module are ssl. Note that overriding the check while loading an incompatible module can cause unpredictable behavior. Added in version 4.01.1031325401 20020906.

- **transferlog $file**
  Sets the web server transfer log to use for this transaction, overriding the web server configured `TransferLog` value. Note that this is only possible in the integrated `vhttpd` environment, i.e. `VortexPath`, `EntryScript` or `ExitScript`. If the script is running in CGI mode (e.g. under IIS or Apache), this setting fails because Vortex has no knowledge of or access to external-server configs. The setting also fails if the `AllowLogFileOverride` setting (p. 591) in the `vhttpd` config is false. Can be used for fine-grained control of transfer log, i.e. to split logs based on different scripts or functions, e.g. in an `EntryScript` (p. 593). An empty string for $file means no log (i.e. `/dev/null`). If $file has no dir component, it is relative to `LogDir`. Otherwise, if it is not absolute, it is `ServerRoot`-relative. Returns 1 if success, 0 on error. Added in version 5.01.1170123063 20070129.

- **errorlog $file**
  Sets the web server error log (not Vortex log) to use for this transaction, overriding the web server configured `ErrorLog` value. Note that this is only possible in the integrated `vhttpd` environment, i.e. `VortexPath`, `EntryScript` or `ExitScript`. If the script is running in CGI mode (e.g. under Windows or Apache), this setting fails because Vortex has no knowledge of or access to external-server configs. The setting also fails if the `AllowLogFileOverride` setting (p. 591) in the `vhttpd` config is false. Can be used for fine-grained control of error log, i.e. to split logs based on different scripts or functions, e.g. in an `EntryScript` (p. 593). An empty string for $file means no log (i.e. `/dev/null`). If $file has no dir component, it is relative to `LogDir`. Otherwise, if it is not absolute, it is `ServerRoot`-relative. Returns 1 if success, 0 on error. Added in version 5.01.1170123063 20070129.

- **applylicense $data $user $pass [$remoteUrl]**
  Applies a Texis license update via the Texis Monitor schedule/license server. May be used by the Webinator admin GUI to upgrade the license. $data contains data in `license.upd` format – typically obtained from Thunderstone tech support – and $user/$pass are the user/password to authorize the license update – typically the administrative account created at installation. The Texis Monitor will verify the user/pass against the `[License Update]` settings in `texis.ini` and apply the license. By default, the license will be applied to the local schedule/license server address configured in the `[Scheduler]` section (e.g. `127.0.0.1`); if the license should be applied to a different server, a URL can be specified with $remoteUrl (this server must be configured to accept outside license update requests).

  Note that most current `<urlcp>` settings apply, so changing settings from the default could result in an insecure transaction. E.g. tracing should be turned off, `secure` set to `required` (if it was changed), etc. See discussion of `[License Update]` settings in the Texis manual for details on
1.5. **BUILTIN FUNCTIONS**

requirements and caveats. Added in version 6. See also the --apply-license command-line option, p. 568.

The return value is a string with a result-code token followed by a human-readable error message, in the form "Err=token Message text here". On success, the return value should start with "Err=Ok". Some other possible tokens include:

- **FetchError**: A network/fetch error occurred.
- **LicenseServiceDisabled**: The Texis Monitor does not have the [License Update] service enabled.
- **SecureConnectionRequired**: A secure (SSL) connection is required but could not be established.
- **Unauthorized**: The transaction was unauthorized, e.g. the user or password was invalid.
- **InvalidLicense**: The license supplied was invalid.
- **InternalServerError**: An internal error occurred on the server.

- **maxurllen $sz**
  Sets the maximum URL length for $url or $urlq to the integer value $sz. The default is 512. This limit prevents huge URLs from being accidentally produced, e.g. if a large or large-number-of-values variable is URL-exported. Exceeding the limit may generate a message such as State URL exceeds limit of 512 bytes: truncated or State query string exceeds limit of 512 bytes: truncated at value 0 of variable $query, and state information will be lost. Added in version 5.01.1237486600 20090319. Returns previous value. Note that the limit applies to $url and $urlq separately, since they are invoked at separate times. Thus with a maxurllen of 512 bytes, a combined $url/$urlq URL could reach 1024 bytes before truncation.

- **tracevortex log $file**
  Sets the Trace Vortex log file. The default is the Vortex script path (without extension), with .vstrace appended. Setting it to empty restores the default value. Returns the previous value. See the Trace Vortex appendix (p. 609) for details on tracing Vortex scripts.

- **tracevortex on|off**
  Turns Trace Vortex logging on or off. See the Trace Vortex appendix (p. 609) for details on tracing Vortex scripts.

- **tracevortex forceflush on|off**
  When Trace Vortex logging is on, force a flush to the log file after every write. This may help ensure the current (or most recent) statement is written to the log if the Texis executable hangs or ABENDs during tracing. Note that it is highly inefficient to enable such constant flushing, and so forceflush should not be enabled normally.

- **compatibilityversion $version**
  Sets Texis compatibility version to $version, which is a Texis version string of the form "major[,minor[,release]]", where major is a major version integer, minor is a minor version integer, and release is a release integer (i.e. the format used in texis -version). This will attempt to make Texis compatible with the given version – i.e. change behavior to match it – so that scripts written for an earlier Texis version might be able to run with the current Texis version. Note
that not all earlier versions are supported, and even if supported, not all behavior changes may be
rolled back. Also, as this is a run-time setting, properties and settings initialized from texis.ini
may be changed. A version that is supported currently may not always be supported in the future,
though generally at least one major version prior to the current one will be; e.g. version 7 supports
setting compatibility to 6. As this setting may have many side effects and change many other settings,
it should generally only ever be called once, at the start of a script, and before any other settings.
Returns “Ok” on success, otherwise a human-readable error message. Added in version 7. See also
the --compatibility-version command-line option.

In version 7, setting compatibilityversion to 6 causes the following changes to restore
version 6 behavior. For details on these and other version 7 changes, see the Texis 7 Features and
Changes appendix (p. 703):

- `inmode` is set to `intersect`
- `strlst` relational operators (less-than, greater-than etc.) with `varchar` operand does not
  promote the `varchar` to `strlst` via `create` mode
- `strlst` comparisons do not use `stringcomparemode`
- Multi-item (`strlst`) index results are not de-duplicated
- `multivaluetomultirow` defaults to `off`
- `<sum>` behavior determined by `args` not format
- `<sql` output="xml"> parses HTML entities
- The default Metamorph index type is `non-inverted` (wordpositions off)
- `varbyte` conversions to/from `varchar` in Texis `hexify/unhexify` bytes
- `varchar_to_strlstsep` is set to `lastchar`, and in future settings the builtindefault
  value becomes equivalent to `lastchar`
- Converting an empty string to `strlst` when `varchar_to_strlstsep` is `create` yields a
  one-empty-string `strlst` (not a zero-item `strlst`)
- The `putmsg` `exceptiononly` state cannot be set
- `<getvar>` and `<setvar>` will not issue messages about illegal/unsettable variables
- `<vxcp` `putmsg` value of `exceptiononly` added

The following version 7 changes are not rolled back when setting `compatibilityversion` 6:

- Trace Vortex script tracing
- REX does not return redundant empty hits
- `<sandr>` does not infinite-loop, and matches like `<rex>`
- `<fmt` `%1/,%1:` codes changed
- Empty-element Vortex statement tags allowed
- Vortex pragma stacks
- `<write` output, `flags`, `skiponfail` options added
- `<vxcp` `timeouttext`, `compatibilityversion` added
- HTTP cookie acceptance relaxed
1.5. BUILTIN FUNCTIONS

- xmlTreeXPathSetContext() added to XML API
- $ret.size is int64 on all platforms
- New SQL operators SUBSET and INTERSECT added
- ALTER INDEX, CREATE INDEX ... WITH options SQL syntax added

- vardebugprintstyle always|once

  Print CSS style for $? variable debug syntax always (i.e. whenever the syntax is used), or only once after first use. Returns 0 on error.

DIAGNOSTICS

The vxcp function returns setting-dependent value(s).

EXAMPLE

```xml
<A NAME=dosomething PRIVATE>
<LOCAL prev>
  <vxcp putmsg all off>
  <$prev = $ret>
  <FunctionThatCausesErrors>
  <vxcp putmsg all $prev>
</A>
```

This function completely disables any messages generated by `<FunctionThatCausesErrors>` that we do not care about, avoiding clutter in the log. However, we do care about errors from other functions in the script, so the previous putmsg action settings are saved and restored.

CAVEATS

The vxcp function was added in version 3.0.947100000 20000105.

It is inadvisable to set a large stack limit unless a script explicitly needs to make use of deep recursion.

Turning signals off with trap can cause aberrant behavior.

SEE ALSO

sqlcp, apicp, urlcp, PUTMSG, TIMEOUT, STACK, TRAP
1.5.70 vxinfo – get miscellaneous Vortex information

SYNOPSIS

<vxinfo $what [$arg ...]>

DESCRIPTION

The vxinfo function returns miscellaneous Vortex information, depending on the $what parameter:

- **outputsz**
  Returns the number of bytes output by the script so far, as a long. This does not count redirected output, i.e. the input to CAPTURE, EXEC etc. blocks.

- **db**
  Returns the current database path used by SQL statements. This is changeable with the DB statement (p. 37). Note: the database used by SQL statements can also be set on a per-statement basis by the DB option to SQL (p. 27).

- **globaldb**
  Returns the global database path, which is used for the EXPORT TABLE variables, and is the default database for SQL statements unless overridden at run-time. This is settable with the DB directive (p. 37).

- **installdir**
  Returns the directory that Texis was installed in. This is usually “/usr/local/morph3” under Unix, or “c:\morph3” under Windows, but may be a different directory depending on the user’s installation. Added in version 3.01.982700000 20010220.

- **executabledir**
  Returns the Texis executables directory, i.e. typically “/usr/local/morph3/bin” under Unix or “c:\morph3” under Windows. Added in version 5.01.1270583600 20100406.

- **htmlmode**
  Returns 1 if in HTML mode, 0 if not. If in HTML mode, variables are automatically HTML-escaped when printed; this is the default if the URL file extension (p. 8) or output MIME type (via a Content-Type header, p. 144) indicates HTML or an HTML-like MIME type. In version 7.01.1390866000 20140127 and later, the MIME types text/xml, application/xml and application/xhtml+xml are also recognized HTML-like MIME types; previous versions only recognized text/html and text/vnd.wap.wml. Can be changed with <vxcp htmlmode> (p. 284). Added in version 3.01.982400000 20100217.

- **platform**
  Returns the platform ID string: the string printed in parentheses by texis -version.

- **features**
  Returns a list of platform-dependent features supported on the current platform. Currently the list may contain zero or more of:
1.5. BUILTIN FUNCTIONS

– RE2 if RE2 regular expressions (p. 151) are supported in REX

Added in version 7.06. See also the version and release parameters to <vxinfo>, and the SQL function hasFeature().

• version [feature]
  Returns the version string: the second line of information printed by texis -version. If feature is given, returns the version string for that feature, which must be one of the tokens returned by <vxinfo features>. Added in version 3.01.967148000 20000824. The $feature argument was added in version 7.06.1468274000 20160711. See also the release and features parameters to <vxinfo>.

• release [feature]
  Returns the release date of the Vortex (texis/vhttpd) executable, or of the $feature (if given), which must be one of the tokens returned by <vxinfo features>. Added in version 3.01.980000000 20010120. The $feature argument was added in version 7.06.1468274000 20160711. See also the version and features parameters to <vxinfo>.

• license $name
  Returns the Texis license value corresponding to $name, which can be one of the following: violationmsg, violationtime, gentime, expiretime, verifytime, verifytrytime, serial, curhits, curhitstime, maxhits, curtblrows, maxtblrows, curtblsz, maxtblsz, curdbrows, maxdbrows, curdboz, maxdboz, curtotrows, maxtotrows, curtotsz, maxtotsz, texismonitorpid, equivpath, uequivpath, vortexlog, defaultdb, defaultscript, vortexflags, texisflags, schemas. Added in version 3.01.985300000 20010322. In version 5.01.1242246000 20090513, the values inittime, maxversion, prevhits, highhits and flags were added. In version 6, the following values were added: fetchesToday, fetchesTodayStart (when fetchesToday count started), fetchesThisMinute, fetchRate3, fetchRate15, fetchRate60 (fetches per minute averaged over the last 3, 15 and 60 minutes), maxfetches (licensed fetch limit per day; 0 is unlimited), metamorphsToday, metamorphsTodayStart, metamorphsThisMinute, metamorphRate3, metamorphRate15, metamorphRate60, maxmetamorphs (license limit), and thisMinuteStart (when the current minute for ...thisMinute values started; if more than one minute ago, such values may be invalid). Only user-initiated, network fetches are counted (e.g. internal system fetches, format-only calls such as <fetch theUrl theData>, and automatically-fetched sub-objects such as frames and scripts, are not counted). In version 6.00.1294878413 20110112, maxversionnum was added.

• vortexlog
  Returns the current Vortex log path, or “-” if standard error. Added in version 5.01.1156453000 20060824.

• content
  Returns the raw content of the POST or multi-part MIME upload to the script, if any. This can be used to proxy the same POST vars to a remote script with submit. (Note that Vortex parses the POST/MIME input for variables and automatically sets them in Vortex; obtaining the raw content is only needed for re-submitting an identical request to another script, or cache systems, etc.) Added in version 3.01.990000000 20010524. Note that multi-part MIME content may be altered (contain nulls) due to memory conservation during the potentially-large upload.
• **objectpath**
  Returns the file path to the currently running object file (e.g. .vtx file). Added in version 6.00.1292382279 20101214.

• **cmdlnargs**
  Vortex command-line arguments, starting with the executable. Added in version 7.01.1394752000 20140313.

• **scriptargs**
  Script command-line arguments. These are arguments after the script on the command line, and thus are only passable and available if `-R` is used (p. 568). If the script is a Unix self-executing “shebang” script, these are the arguments after the “executable” script. Added in version 7.01.1394752000 20140313.

• **stack dump [$N]***
  Dumps current Vortex stack information and returns it in $ret, one level per line. Optional $N, which defaults to 0, is bitwise OR of flags:
  
  - 1 Dump all items, not just function calls
  - 2 Dump parameters too
  - 4 Dump full length of parameter values
  - 8 Print global variables too
  
  This information can be used when debugging a script, to determine where a function was called from. These flags, and the format of the output, are subject to change without notice in future versions. Added in version 3.01.987820000 20010420.

• **texisconf [$section [$setting [$default]]]***
  Returns current texis.ini configuration value for $setting in $section. The $section may be numeric, where sections are numbered from 0 in the order they appear in the file; this is useful if the same-named section occurs multiple times (e.g. Httpd Fast CGI). If $default is given, that value is returned as a default value if the setting is not present. If $setting is not given, all setting names in $section are returned. If $section is not given, all section names in the file are returned, in file order. Section and setting names are case- and space-insensitive. Added in version 4.04.1079750000 20040319.

• **texisconffrom $file [$section [$setting [$default]]]***
  Same as texisconf, but reads settings from $file instead of the active config. If $file is empty, the default texis.ini file in the install dir is used (this may still be different from the file used by texisconf, i.e. if `-conf` overrode the config file on the command line, the latter will be used by texisconf). Added in version 4.04.1079750000 20040319.

• **texisconffile**
  Returns the file path to the texis.ini file used by the current invocation of Vortex. Note that the file may not exist, in which case it is the file that would be used were it to exist. Added in version 5.01.1270688277 20100407.

• **sourcepath**
1.5. BUILTIN FUNCTIONS

Returns the file path to the currently running script (not module) source, including extension if applicable. More permanent than the $sourcepath variable, which could be accidentally modified by the script. Added in version 4.04.1077500000 20040222.

- sourcepathnoext
  Same as sourcepath, but without extension. Added in version 5.01.1197080000 20071207.

- scriptroot
  Returns the full file path to the effective Script Root, i.e. after obtaining texis.ini, conf/vhttpd.conf or default value (as appropriate via precedence) and expanding variables. May be empty if Script Root is not applicable or computable in the current invocation, i.e. if the script was run from the command line. Added in version 5.01.1268365000 20100311.

- execreadline
  When called from within an <EXEC> block, returns the next line of data from the program’s output, without the trailing newline, blocking if needed until data is available. Returns empty (0 values) if EOF has been reached, i.e. the program exited and all data has been read. Each line read is removed from the (later) $ret return value of the closing </EXEC>.

  Normally it is easier to simply use $ret after </EXEC>. However, at </EXEC> the program’s standard input has been closed and the program has exited. <vxinfo execreadline> provides a way to read data in real time from a program, while still potentially sending new input to it. This is useful for “interactive” programs, where the next line of input to the program may depend on previous output from it. Added in version 5.01.1109710555 20050301.

- compatibilityversion
  Returns a double value for the current Texis compatibility version, i.e. the major.minor version Texis is attempting to emulate. The default (if no <vxcp compatibilityversion> set) is the Texis version number.

- timeout
  Returns the current Vortex script timeout, as a date deadline, or -1 long if no timeout. Added in version 5.01.1148484000 20060524.

- tracevortex
  Returns 1 if Trace Vortex (p. 609) is active, 0 if not.

- tracevortex log
  Returns the Trace Vortex log file path (p. 289).

- tracevortex forceflush
  Returns 1 if Trace Vortex force-flushing (p. 289) is enabled, 0 if not.

DIAGNOSTICS

The vxinfo function returns option-dependent value(s).
EXAMPLE
This example logs the path and size of a page after it is delivered by the script:

... print a page of results ...
<vxinfo outputpsz>
<SQL NOVARS "insert into log values($REQUEST_PATH, $ret)"/></SQL>
<exit>

CAVEATS
The \texttt{vxinfo} function was added in version 3.01.967200000 20000824.

SEE ALSO
\texttt{vxcp}
1.5. BUILTIN FUNCTIONS

1.5.71 hash – produce a hash or checksum for data

SYNOPSIS

<hash [options ...] [DATA=$buf[ ]]> or

<hash [options ...]> ... </hash>

DESCRIPTION

The hash function produces a hash or checksum value of the data $buf (or block output), using a specified algorithm. The value returned – a small-size type, usually counter – is fairly unique to the given data, i.e. it is very unlikely that a different input data value will have the same hash value. The hash value can be used as a fast way of comparing large data fields: if the hash values are the same, the data fields are probably identical. This can replace a large number of very long string compares with much faster counter (or other compact type) compares. Hash values are also used to verify the integrity of transmitted data: if the hash computed by the receiver does not match the value transmitted, the data is corrupted.

With no options, hash computes a gw (i.e. Webinator) style hash of $buf, returned as a counter value. In Vortex version 4.04.1066340000 20031016 and later, new hash types and options are available. No further options are available in earlier versions (including the block-function version).

If the DATA parameter is not supplied (either explicitly named, or as the last unnamed parameter), then the hash function becomes a block function, and the input is taken from the output of the enclosed Vortex code, rather than the $buf parameter. This is useful for producing a checksum of the output of part of a script, without having to first save the (possibly large) output value in a variable first (see examples).

Options are as follows:

- TYPE=$type
  Specifies what type of hash to produce (i.e. the algorithm). Also gives a default return variable type (see RET option), as different algorithms produce different native data types. The possible values for $type, the PROVTYPE which supports them (need not be given; see below), and default RET types are:

  - gw (supported by PROVTYPE=texis)
    A gw (Webinator) style hash. This is the default if TYPE is unspecified or empty. The default RET type for gw is counter, which returns a cross-platform-compatible value. Other RET types may truncate or return different values across platforms for gw.

  - crc32 (supported by PROVTYPE=texis)
    A CRC32-compatible 32-bit checksum. The default RET type is long, which returns a cross-platform-compatible value. Added in version 4.04.1083100000 20040427.
- **adler32** (supported by PROVTYPE=tesxis)

  An Adler32-compatible 32-bit checksum (similar to crc32 but slightly faster to compute, though incompatible). The default RET type is long, which returns a cross-platform-compatible value. Added in version 4.04.1083100000 20040427.

- An OpenSSL digest name (supported by PROVTYPE=openssl)

  An OpenSSL algorithm, such as md5, md2, sha1, dss1, DSA-SHA1-old, RSA-SHA1-2, etc. The default RET type is hex, which returns a cross-platform-compatible value. Consult an OpenSSL manual (online at http://www.openssl.org/) for details on algorithms.

- A Microsoft CryptoAPI algorithm (supported by PROVTYPE=PROV_...)

  A Microsoft CALG_... algorithm, such as:

  ```
  CALG_3DES CALG_RC4
  CALG_3DES_112 CALG_RC5
  CALG_AGREEDKEY_ANY CALG_RSA_KEYX
  CALG_CYLINK_MEK CALG_RSA_SIGN
  CALG_DES CALG_SCHANNEL_ENC_KEY
  CALG_DH_EPHEM CALG_SCHANNEL_MAC_KEY
  CALG_DH_SF CALG_SCHANNEL_MASTER_HASH
  CALG_DSS_SIGN CALG_SEAL
  CALG_HMAC CALG_SHA
  CALG_HUGHES_MD5 CALG_SHA1
  CALG_KEYA_KEYX CALG_SKIPJACK
  CALG_MAC CALG_SSL2_MASTER
  CALG_MD2 CALG_SSL3_MASTER
  CALG_MD4 CALG_SSL3_SHAMD5
  CALG_MD5 CALG_TEK
  CALG_PCT1_MASTER CALG_TLS1_MASTER
  CALG_RC2
  ```

  Or an integer numeric value for such a CALG_... algorithm may be given. The default RET type is hex, which returns a cross-platform-compatible value. Consult a Microsoft SDK manual for details on algorithms.

- **OUTPUT=$what**

  Specifies what to output (print). The default (or if an empty string is given) is none, i.e. nothing, because the hash value is returned in $ret. See RET for a list of the possible values for $what. This option can be used to re-print the input of a block function at the same time the hash is being returned in $ret (see examples).

- **RET=$what**

  Specifies the value and variable type to return in $ret. The value $what may be one of:

  - **bin** - the hash as a varbyte value
  - **counter** - the hash as a single big-endian counter value
  - **hex** - the hash as a hexadecimal varchar value
  - **input** - the input data (same type), instead of the hash value; useful for OUTPUT option
  - **long** - the hash as a single big-endian long value (added in version 4.04.1083100000 20040427)
Note: not all return types are compatible with all hash types, e.g. a `counter` value may truncate an MD5 hash on some platforms. The default return type (if unspecified or empty) is determined by what hash type is selected. See the list under the `TYPE` option for more information.

Note that numeric return types (e.g. “counter”, “long”) are big-endian (first byte of hash is most-significant in the returned number) for consistency, regardless of whether the platform is big- or little-endian. This helps ensure portability: two platforms with the same-size `counter` and `long` types will produce the same hash values for those types (for the same input and `TYPE`), regardless of platform endian-ness.

- `DATA=$buf`
  Specifies the input data value(s) to produce a hash of. If this option is given, the function is a non-block-function call, i.e. no ending `<hash>` is expected, and a hash is returned for each corresponding value of `$buf`. If this option is missing, the function is a block call, and input is taken from the output produced by the script up to the next matching required `<hash>` tag. Note that for back-compatibility, an unnamed parameter is taken as the `DATA` parameter (and must be the last option).

- `PROVTYPE=$provtype`
  This parameter can be left unspecified; a default is usually correctly determined. It specifies the type of provider to use (and indirectly, the API). Along with `PROVIDER` and `TYPE`, this determines the exact algorithm, padding scheme, key length etc. to use to produce the hash or checksum. `PROVTYPE` may be `texis` for the Texis API, `openssl` for OpenSSL API, or on Windows platforms, any of the following Microsoft CryptoAPI `PROV_...` provider types (not all supported on all versions of Windows):

  ```
  PROV_DH_SCHANNEL  PROV_MS_EXCHANGE
  PROV_DSS          PROV_RSA_AES
  PROV_DSS_DH       PROV_RSA_FULL
  PROV_EC_ECDSA_FULL PROV_RSA_SCHANNEL
  PROV_EC_ECDSA_SIG PROV_RSA_SIG
  PROV_EC_ecnra_FULL PROV_SPYRUS_LYNS
  PROV_EC_ecnra_SIG PROV_SSL
  PROV_FORTEZZA
  ```

  The default (if `PROVTYPE` is unspecified or empty) is the first provider type that can support the `TYPE` specified, in the priority order `texis,openssl,PROV_RSA_FULL`. E.g. if `TYPE=gw` is specified, `PROVTYPE` defaults to `texis`; if `TYPE=md5` is given, `PROVTYPE` defaults to `openssl` (even though `PROV_RSA_FULL` supports md5 too); if a recognized Microsoft CryptoAPI `CALG_...` algorithm is given for `TYPE`, `PROVTYPE` defaults to `PROV_RSA_FULL`.

- `PROVIDER=$provider`
  This parameter can be left unspecified; a default is usually correctly determined. It specifies the engine or cryptographic provider to use, depending on the value of `PROVTYPE`:

  - `PROVTYPE=texis`
    `PROVIDER` must be (and the default is) `texis`. 
- **PROVTYPE=openssl**
  PROVIDER specifies the engine to use. The value must be a registered OpenSSL engine; the default if unspecified or empty is provided by the OpenSSL API.

- **PROVTYPE=PROV_... value (Microsoft CryptoAPI)**
  PROVIDER specifies the cryptographic provider to use. It may be one of the following values:
  * MS_DEF_DSS_DH_PROV
  * MS_DEF_DSS_PROV
  * MS_DEF_PROV
  * MS_DEF_RSA_SCHANNEL_PROV
  * MS_DEF_RSA_SIG_PROV
  * MS_ENHANCED_PROV
  * MS_ENHANCED_RSA_SCHANNEL_PROV
  * Microsoft Base Cryptographic Provider v1.0
  * Microsoft Enhanced Cryptographic Provider v1.0
  * Microsoft RSA Signature Cryptographic Provider
  * Microsoft Base RSA SChannel Cryptographic Provider
  * Microsoft Enhanced RSA SChannel Cryptographic Provider
  * Microsoft Base DSS Cryptographic Provider
  * Microsoft Base DSS and Diffie-Hellman Cryptographic Provider

  Or PROVIDER may be another provider name recognized by CryptoAPI. The default if unspecified or empty is provided by the CryptoAPI.

**DIAGNOSTICS**

The hash function returns a hash or checksum value for each value of DATA=$buf (or the output of the enclosed block). The return type varies with options but defaults to counter.

**EXAMPLE**

```<%theurl = "http://some.host.com/">
<fetch $theurl>
<urlinfo text>
<$page = $ret>
<hash $page>
<$hash = $ret>
<SQL "select Hash from pages where Hash=$hash"></SQL>
<IF $loop gt 0>
  Page already seen elsewhere.
<ELSE>
  <SQL "insert into pages values($theurl, $page, $hash)"></SQL>
</IF>
```
In the above example, a table of fetched HTML pages is kept, along with a hash value for each page (and an index on the hash field). The hash values are used as a quick way of keeping only unique pages. If multiple URLs being fetched point to the same page, only one copy of the page is desired. To do this, the hash values are compared: if the hash value of a newly fetched page is already present in the table, it’s virtually certain that page was already fetched. Since no options are given to `<hash>`, a gw-style (Webinator) hash is used.

Comparing hash values is far faster than attempting to compare each page’s text itself, since the hash is just a small fixed size value (counter in this case) whereas the page text may be megabytes long.

```
<hash TYPE=md5 OUTPUT=input>
   Start of message
   <SendLargeLogFile>
   End of message
</hash>
MD5-Checksum=$ret
```

In the above example, the block-function mode of hash is used to generate an MD5 checksum of the message being printed by the Vortex code enclosed by the block. By using the block mode, we do not have to construct the message and assign it to a variable, which not only saves code but in this case memory too, as `<SendLargeLogFile>` (not shown) would otherwise generate a very large variable. By using the OUTPUT=input option, the message is also printed as it’s checksummed, so we do not lose it.

CAVEATS

The hash function was added May 16 1997. New options such as block mode were added in version 4.04.1066340000 20031016 and 4.04.1083100000 20040427.

The OpenSSL algorithms depend on the OpenSSL plugin; this is provided with all versions of Vortex and is loaded automatically when needed, but does use on external libraries (i.e. which may be missing in a broken install).

The Microsoft CryptoAPI algorithms are only available on Windows versions of Vortex.

It is possible, though unlikely, that two distinct input data values will have the same hash value. The probability of such a hash collision varies with the algorithm (TYPE) used; md5 is currently one of the best (i.e. lowest probability).

SEE ALSO

fetch, urlinfo
SYNOPSIS

<geo2code $lat $lon [$radius]>

DESCRIPTION

**Note:** geo2code is a deprecated function. Use the SQL functions `latlon2geocode()` and `latlon2geocodebox()` where possible instead.

The `geo2code` function encodes each latitude/longitude coordinate given into one integer. This number can be indexed and used with a special variant of Texis' `between` operator for bounded-box searches of a geographical region. The `$lat`, `$lon` and optional `$radius` parameters are integers in the form `DDDM M SS` (DMS “degrees minutes seconds” format), with negative numbers representing south latitudes and east longitudes respectively (note that longitude signs are the opposite of ISO 6709, which is east-positive).

Valid values for `$lat` are -90 to 90 degrees inclusive (i.e. -900000 to 900000). Valid values for `$lon` are -360 to 360 degrees inclusive. A `$lon` value less than -180 degrees will have 360 degrees added to it, and a `$lon` value greater than 180 degrees will have 360 degrees subtracted from it. This allows longitude values to continue to increase or decrease when crossing the International Dateline, and thus avoid a non-linear “step function”. Passing invalid `$lat` or `$lon` values will return $-1$. These changes were added in version 5.01.119395804 20071101.

By using this encoding method and a special variant of the `between` operator, only one field need be indexed and searched, instead of both latitude and longitude separately, which can be time-consuming at search. The encoded position can be decoded back to latitude/longitude with the `code2geo` function (p. 304).

If the optional `$radius` parameter is given, then a pair of numbers is returned for each coordinate. The pair represents the encoding for the NW and SE corners of a box that is twice the `$radius` in length on a side, with the given coordinate at the center. The pair is returned as a comma-separated parenthetical string. The result can thus be directly inserted into a SQL statement to search the geographical region within `$radius DDDMMSS` of the given coordinate (DMS “degrees minutes seconds” format). Where available, the `latlon2geocodearea()` SQL function is preferred however, as it returns a true parameter, not SQL statement fragment.

DIAGNOSTICS

`geo2code` returns a `long` integer for each latitude/longitude coordinate pair given, for a `BETWEEN (aaa, bbb)` GIS search. If a `$radius` argument is given, `geo2code` returns a comma-separated parenthetical pair of numbers instead, for the bounding box that contains a circle of that
radius centered on the point. All arguments are Texis/Vortex DDDMMSS integers (DMS "degrees minutes seconds" format, west-positive).

EXAMPLE

In this example, latitude/longitude positions for cities were encoded with geo2code into the GeoCode field of a table. A $latitude and $longitude are passed to this function (e.g. from a form) and used to search for cities in a boxed geographic region centered on the given coordinate:

```html
<A NAME=search>
Cities within the region:
<geo2code $latitude $longitude 20000> <!-- +/- 2 degrees -->
<SQL "select Name from city where GeoCode between " $ret>
 $Name
</SQL>
</A>
```

CAVEATS

The geo2code function was added in version 2.1.904800000 19980902.

The returned value in $ret from geo2code is platform-dependent in format and accuracy; it should not be copied across platforms. On 32-bit-long platforms it is accurate to 32 seconds (about half a mile). On 64-bit-long platforms it is accurate to 1 second (about 100 feet or less).

The between operator used here should have a regular (B-tree) index on the encoded (left-side) field, and use the parenthetical syntax. A search without parentheses (e.g. "between $x and $y") loses the special interpretation of the values and will not work correctly with geo2code values.

Note that the Texis SQL functions latlon2geocode() etc. default to east-positive longitudes, not west-positive as Vortex <geo2code> does.

SEE ALSO

<code2geo> Vortex function, latlon2geocode(), latlong2geocodearea() SQL functions
1.5.73  code2geo – decode latitude/longitude encoding

SYNOPSIS

<code2geo $num>

DESCRIPTION

Note: code2geo is a deprecated function. Use the SQL functions geocode2lat() and geocode2lon() where possible instead.

The code2geo function decodes a geo2code-generated encoding into a latitude/longitude pair, returned as a space-separated pair of numbers in a string. The returned coordinate is in the Texis/Vortex $DDDMMS$ form (DMS “degrees minutes seconds” format, west-positive).

DIAGNOSTICS

code2geo returns a string, the space-separated latitude and longitude for the given geo2code-encoded number.

EXAMPLE

<A NAME=list>
   Cities and their positions:
   <SQL "select Name, GeoCode from city">
      <code2geo $GeoCode>
         $Name $ret
   </SQL>
</A>

CAVEATS

The code2geo function was added in version 2.1.904800000 19980902.

As with geo2code, since the encoded value is platform-dependent in accuracy and format, it should not be copied across platforms, and the returned coordinates from code2geo may differ up to about half a minute from the original coordinates.

SEE ALSO

<geo2code> Vortex function, geocode2lat(), geocode2lon() SQL functions
1.5.74 pdfxml – convert Metamorph hit to PDF markup information

SYNOPSIS

<pdfxml $query $Body [options ...]>}

DESCRIPTION

The pdfxml function enables Metamorph hit markup to be viewed in a PDF (Adobe Acrobat) file. It executes the Metamorph $query on a PDF document ($Body) and returns the information needed by a user’s PDF browser plugin to mark up the resulting hits.

Note that $Body must be the exact text returned by the PDF plugin for Thunderstone’s Webinator or Texit. If it is modified in any way, character and word counts may be off, causing incorrect highlighting information to be sent to the browser.

Options that may be specified are:

- **$color**
  The color to show highlighted terms in. Must be a color specified in RGB of the form #RRGGBB (by Adobe specification).

- **words**
  Flag that indicates that the hit markup should use word mode. Mutually-exclusive with characters mode. This is the default, and should be used for anytotx plugins that use the “Adobe Acrobat TK” library (anytotx --identify does not show a pdf: version; prior to version 4.02.1038324681 20021126). Added in version 4.00.999800000 20010906.

- **characters**
  Flag that indicates that the hit markup should use character mode. Mutually-exclusive with words mode. This mode should be used with anytotx plugins that use the XPDF library (anytotx --identify shows a pdf: version; after version 4.02.1038324681 20021126). Added in version 4.00.999800000 20010906.

- **active**
  Indicates that the browser’s PDF viewer should jump to the first match upon displaying the document. Mutually-exclusive with passive. This is the default. Added in version 4.00.999800000 20010906.

- **passive**
  Indicates that the browser’s PDF viewer should not jump to the first match upon displaying the document. Mutually-exclusive with active. Added in version 4.00.999800000 20010906.

- **showhits**
  Indicates that the matching terms should be included in the XML output as comments. This is primarily for debugging purposes and should generally not be used in a production environment. Added in version 4.00.999800000 20010906.
• **startpage $pg** or **startpg $pg**
  Specifies the page number that the $Body document actually starts at. The default is 0 (the first page), i.e. $Body is the complete document. This option is used to keep the browser plugin in sync when partial-document $Body arguments (e.g. single pages) are used. For example, if $Body actually starts at the third page of the original document, use **startpg 2**. Added in version 5.00.1092761457 20040817.

• **charset $charset**
  Specifies the character set of $Body. The default is ISO-8859-1. Multi-byte character sets such as UTF-8 can cause erroneous highlighting offsets if the character set is not specified with this option. Note that this is the character set of $Body, not necessarily that of the original PDF. Added in version 5.01.1104778576 20050103.

**DIAGNOSTICS**

The pdfxml function returns a list of strings to be sent to the Web browser's PDF viewer plugin.

**EXAMPLE**

```xml
<EXPORT $query>
<EXPORT $id>

<A NAME=xml>
<SQL MAX=1 "select Body from html where id = $id">
<pdfxml $query $Body="#00FF00">  
<LOOP $ret>
  $ret
</LOOP>
</SQL>
</A>

<A NAME=main>
...

<SQL "select Url, Title, id from html where Title\Body like $query">
<substr $Title 0 14>
<IF $ret eq "PDF Document (">
  <CAPTURE>#xml=http://$HTTP_HOST$url/xml.txt</CAPTURE>
<ELSE>
  <$ret = "">
</IF>
<A HREF="http://$Url$ret">$Title</A>
</SQL>
```
In this example, the main function excerpt prints the URLs for documents matching $query$, from a Webinator html table. For most documents this is http:// plus the $Url. For PDF documents however, an anchor is attached (#xml=...) that contains the URL to hit markup information. If the user’s Web browser is configured with a PDF viewer, the viewer will fetch this hit information URL (which points to the xml function here) and use it to mark up the PDF document.

CAVEATS

The pdfxml function was added in version 2.1.864700000 19970527.

The PDF (anytotx) plugin for Thunderstone’s Webinator must be used to generate the $Body value passed to pdfxml. Also, the web user’s browser must have a configured PDF viewer to fetch and use the PDF markup information.

The XML generated by this function conforms to Adobe’s “Highlight File Format” specified in Adobe Technical Note #5172. It does not necessarily conform to any XML “standard”.
1.5.75  

**xtree – maintain sorted list of values**

**SYNOPSIS**

```xml
<xtree [XTREE=$treename] $command [$values] [$treename][ /]> 
```

or

```xml
<xtree [XTREE=$treename] [options] $command [$values] [$treename]> 
... 
</xtree>
```

**DESCRIPTION**

The `xtree` function is a fast way to maintain a sorted list of temporary string values in memory – faster than using a SQL table. Values can be added, removed, counted and deleted from a tree, named with `$treename` (default tree name is empty). The `$values` parameter is the list of values to operate on, for `INSERT`, `DELETE` and `SEARCH`; for `SET` it is the option(s) to set. Otherwise the `$values` parameter is not given, or is empty if `treename` is specified via the last-argument syntax.

Each value in an xtree has a count and a sequence number associated with it. The count is the number of times the value was inserted. The sequence number is the order in which the value was first inserted, starting with 0 for the first value inserted. These values are retrieved either with the `COUNT` and `SEQ` commands (non-loop syntax), or in the `$ret.count` and `$ret.seq` variables (looping version).

If no options (or only `XTREE` option) are given to `xtree`, it returns values in `$ret` as a simple function.

If any of the following options are given (except `XTREE`), then `xtree` is a looping block command, e.g. a closing `"</xtree>"` tag is expected, and `$loop`/$`next` are set as in `SQL` (`$loop` always starts at 0). Any statements inside the block are executed once per returned value, with `$ret` being a loop variable accumulating values. In addition, the variables `$ret.count` and `$ret.seq` are also looped over and set to the corresponding count and sequence numbers for `$ret`. The loop can be exited with `BREAK` or `RETURN`. The loop syntax was added in version 2.6.938200000 19990924. Options are:

- **ROW**
  As in `SQL`, do not accumulate hits in `$ret/$ret.count/$ret.seq`, and do not make them loop variables; each new value erases the previous. `ROW` should be used in a looping `xtree` when a large number of return values are expected but only need to be examined one at a time; this saves memory and time since all the values do not have to be stored in memory. `ROW` should also be used when functions are called within the block, because otherwise `$ret` is a loop variable, hindering multi-value returns. `ROW` should also be used when the `COUNT` and `SEQ` commands are not to be used, as it saves the memory of preserving those values for a later `COUNT` or `SEQ`.

- **SKIP=$n**
  Skip the first $n values. This does not affect the value of `$loop`. 
1.5. BUILTIN FUNCTIONS

- **MAX=$n**
  
  Return at most $n values.

- **XTREE=$treename**
  
  Use tree named $treename instead of the default. This option was added in version 5.01.1217380000 20080729. The old syntax was to give the tree name as the last argument, with no option name, which can be confusing. Only one syntax – XTREE option or last-unnamed-argument – may be used in a statement.

The $command argument determines what xtreet does, whether it takes a $values argument or not, and what values, if any, are returned. It is one of the following commands:

- **CLEAR**
  
  Clear the tree. All counts are set to 0, but the tree and its values remain in memory. $values need not be set. No values are returned.

- **DUMP**
  
  Dump the tree. The return value $ret is a unique, sorted list of the values in the tree that have non-zero counts, in ascending order. In the looping syntax, $ret.count and $ret.seq are set to the corresponding count and sequence numbers; in the non-looping syntax, these values must be retrieved with separate COUNT and SEQ commands afterwards. If the CLEAR option (not command) is set with SET, each value is also cleared as it is returned. $values need not be set.

- **COUNT**
  
  Dump the value counts from the immediately previous non-looping DUMP, SEARCH or INSERT command on the same-named tree. $ret will contain a list of the number of times each value from the operation occurs in the tree. For the looping syntax, nothing is returned – the counts and sequences are available during the DUMP/SEARCH in $ret.count and $ret.seq. $values need not be set.

- **SEQ**
  
  Returns the corresponding sequence numbers (starting at 0), from the immediately previous non-looping DUMP, SEARCH or INSERT on the same-named tree. These indicate the order in which the strings were originally added. For the looping syntax, nothing is returned – the counts and sequences are available during the DUMP/SEARCH/INSERT in $ret.count and $ret.seq. $values need not be set.

- **INSERT**
  
  Adds the $values to the tree, if not already present, and increment their counts. If the NOADD option is set (below), new values are not added; only already-present values are incremented. In versions after 3.01.963400000 20000712, the inserted values, counts, and sequence numbers are returned. This saves an extra SEARCH to determine the numbers right after an INSERT. Previous versions returned nothing.

- **DELETE**
  
  Deletes the $values from the tree, i.e. sets their counts to 0. No values are returned in $ret.

- **SEARCH**
Returns a list of the $values that are in the tree. For the looping syntax, $ret.count and $ret.seq are set to the corresponding count and sequence numbers. For the non-looping syntax, use COUNT/SEQ immediately afterward to obtain the counts and sequence numbers.

- **SET**
  Set the option(s) given in $values. In version 6.00.1305253000 20110512 and later, 1 is returned on success, 0 on failure; previous versions always returned empty string. Looping version iterates 0 times (since there is no equivalent count and sequence number for the returned success/fail code).

  - **noadd**
    Do not add new values to the tree during an INSERT. Inserting a previously-cleared value will make it “reappear”, because it was present before but now has a non-zero count. But wholly-new values will not be inserted.

  - **clear**
    Also clear the tree during a DUMP.

  - **stringcomparemode $mode**
    Set string compare mode – how values are compared – for the tree to $mode. The syntax is the same as the apicp setting stringcomparemode (p. 120). The default (and the meaning of values default or builtin) is the current apicp stringcomparemode. This option must be set before any data is inserted into the tree, as it would alter the sort order. Added in version 6.

  - **storefolded $onOff**
    Set the store-folded flag for the tree to $onOff (i.e. “on” or “off”). Enabling store-folded increases speed slightly, at the expense of not retaining inserted values exactly. If storefolded is on, values are folded (according to stringcomparemode) once, at insert time, and stored folded. This increases speed because only one folding is needed. However, values returned from DUMP etc will be the folded value, not the originally-inserted value. E.g. if stringcomparemode is set to ignorecase, the inserted value “McDuff” will be DUMPed as “mcduff”.
    
    If storefolded is off (the default), values are folded whenever needed for comparisons, and stored exactly as inserted. This decreases speed because potentially more than one folding is needed. However, the originally-inserted value is preserved, i.e. if “McDuff” is inserted in an ignorecase tree, it will be DUMPed as “McDuff”.
    
    In either case, stringcomparemode is respected for searches, inserts etc.; just the value of $ret is affected. This option must be set before any data is inserted into the tree, as it would alter the sort order. Added in version 6.

  - **defaults**
    Restores defaults: noadd, clear and storefolded are off; stringcomparemode is the same as the <apicp> value. Note that since the last two settings can only be set on an empty tree, defaults can only be fully restored on an empty tree as well. In version 6.00.1305312600 20110513 and earlier, stringcomparemode and storefolded mode were not restored at all when setting defaults.

- **GET**
  Gets the option(s) listed in $values. Added in version 6.00.1305253000 20110512. Options and what they return are:
1.5. BUILTIN FUNCTIONS

- noadd
  1 if not adding new values to the tree during an INSERT; 0 if adding.

- clear
  1 if clearing the tree during a DUMP, 0 if not.

- stringcomparemode
  The current stringcomparemode for the tree.

- storefolded
  1 if storing values folded, 0 if not.

- numitems
  The number of items in the tree, i.e. the number of non-zero count items (returned by DUMP). It is much faster (constant-time) to obtain this value via get numitems than by DUMPing the tree and checking $loop.

- numallocateditems
  The number of allocated items in the tree, i.e. including zero-count (cleared) items.

- memused
  The number of bytes of memory used by the tree.

- FLUSH
  Remove the tree and free its memory.

DIAGNOSTICS

The xtreet function returns either nothing, a list of strings, or a list of integer counts. If invoked as a looping command, $ret.count and $ret.seq may be set as well.

EXAMPLE

<$txt = "senselessness" "this" "test" "finesse">
<rex "." $txt> <!-- split the words into letters -->
<xtree INSERT $ret>
<xtree ROW DUMP>
 $ret $ret.count
</xtree>

The output would be a count of how many times each letter occurs among all the words:

e 7
f 1
h 1
i 2
l 1
CAVEATS

The xtree function was added Apr. 29 1997. The loop syntax was added in version 2.6.938200000
19990924, as was support for full binary data.

Note that $loop, $next, $ret.count and $ret.seq are only set in the looping version.

It is not possible to modify an xtree while inside an xtree loop for the same tree, nor is it possible to start another DUMP inside the loop. An error message such as “Cannot flush xtree: still in use” or “Cannot insert into xtree while walking it” may be generated in such cases.

In long-running scripts, trees should be FLUSHed when done, to save memory. Since they are kept only in memory, they will be lost when the script exits. ROW should be used whenever possible to save memory and speed.

In version 6 and later, string comparisons use the current apicp stringcomparemode setting (p. 120), changeable with the set stringcomparemode tree option.

SEE ALSO

sort, uniq
1.5. BUILTIN FUNCTIONS

1.5.76 profiler – assist in searching query profiles

SYNOPSIS

<profiler INIT $table [$profile [$field [[$db]]]]>
<profiler GET $msg [$profile]>

DESCRIPTION

The profiler function is helpful in managing a large number of relatively fixed queries against changing data, for example users’ search profiles to match against incoming news data. When a news message arrives, it is desired to find out which of many users’ queries, kept in a table, match the new message. A Metamorph counter index and a LIKEIN query are used for this purpose. However, profiler can first be used to optimize the query string given to LIKEIN, speeding the search. (For more details on Metamorph counter indexes and the LIKEIN operator, see the Texis manual.)

A profile is first initialized with the INIT command, giving it the name of the SQL $table (in database $db – current database if empty/unspecified) with the queries, and an optional $profile name (to distinguish it if multiple profiles are used). The optional $field argument is the field in the table that holds the query for each profile; if it is unspecified or empty the first Metamorph index found on the table will be used to determine the field. INIT loads the words from the index into memory for later use.

When a new message is to be searched against, the GET command is used to intersect its words with those of all the queries. The result is a list of only those words that occur in both the message and at least one query, to be passed to LIKEIN. This is generally smaller than the original message, and thus faster to search.

When the table of queries is modified, profiler will update its internal copy of the index of words, after the Metamorph index is updated, the next time GET is called.

DIAGNOSTICS

The profiler INIT command returns “Ok” on success, otherwise a human-readable error message. The profiler GET command returns the intersection of the words in $msg with those in the index.

EXAMPLE

... create table SearchTable with queries and index ...
<profiler INIT SearchTable testProfile QueryField>
... get new message $msg ...
<profiler GET $msg testProfile> <!-- trim the $msg -->
<SQL "select Name from SearchTable where QueryField likein $ret">
  $Name matches the message
</SQL>
CAVEATS

The `profiler` function was added in version 2.6.938540000 19990928.

The `$db` option was added in version 7.05.1459800000 20160404.

See the Texis manual for details on Metamorph counter indexes and `LIKEIN` searches.

The Metamorph indexes on the query table must be updated for `profiler` to take note and update its memory copy.

Prior to version 7.06.1487025000 20170213, the `INIT` command always returned empty, regardless of success or failure.

SEE ALSO

`xtree`, `wordlist`
1.5.77  sleep – sleep for a while

SYNOPSIS

<sleep $seconds [wake]>

DESCRIPTION

The sleep function pauses (sleeps) for $seconds seconds (for each value of $seconds).

DIAGNOSTICS

sleep returns the number of seconds remaining to be slept (a float), which is normally 0. However, if the "wake" argument is given, sleep may wake up early and return a non-zero value, if a signal was caught. (This can be used to wait for another process to signal the script and wake it up.)

CAVEATS

The sleep function was added Sep. 17 1996. The wake flag was added Sep. 1 2000; previous versions behaved as if it were always set.

Sleep durations with fractions of a second may be given, e.g. 1.5. However, the actual sleep duration may vary depending on the platform’s clock resolution.
1.5.78  sysinfo – get system-specific information

SYNOPSIS

<sysinfo $what [$arg ...]>

DESCRIPTION

The sysinfo function obtains various system- or OS-level parameters. This information can be used to monitor the performance of large or long-running processes. The $what argument determines what is returned:

- **time** [wallclock|continuousfixedrate|continuousvariablerate]
  Given second arg wallclock (or empty/no second arg), returns the current wall-clock (real) time, in seconds since midnight Jan. 1 1970 GMT, as a double. Fractions of a second up to the system clock resolution are returned as well. This is the same value as Unix date displays, but with sub-second resolution. It can be printed and inserted as a Texis date value. Note: User and system time – obtainable with the proctime option – more accurately reflect the CPU utilization of a process. Also, the time needed to fetch a Web page is obtainable with options to urlinfo (p. 185).
  Given second arg continuousfixedrate, the time from a fixed-rate, continuous clock is returned, as a double value of seconds (plus fraction). This clock continuously increments at a fixed rate; the starting point and rate are fixed at boot. Thus it is not affected by forward/backward jumps in the system time (e.g. when the Unix date is set), nor by rate changes in the system time (e.g. from NTP speedup/slowdown). However, this clock’s epoch (zero starting point) is undefined; it is not midnight Jan. 1 1970 GMT, so printing it as a Texis date value will give meaningless year/month/day/etc. values. It should only be used for comparison with other continuousfixedrate times during the same system lifetime (i.e. since boot). It can be used to set deadlines or time events in a manner that is unaffected by system time changes. On platforms where this clock is unsupported, -1 is returned.
  Given second arg continuousvariablerate, a clock similar to continuousfixedrate is returned, except that the rate may vary. I.e. it is subject to the same speedup/slowdown as the system clock, but not the system clock’s jumps; this clock will continuously increment, regardless of date changes. On platforms where this clock is unsupported, -1 is returned.
  The wallclock, continuousfixedrate and continuousvariablerate arguments were added in version 7.06.1510695000 20171114.

- **boottime**
  Returns the system boot time, in seconds (plus fraction) since midnight Jan. 1 1970 GMT. Added in version 7.06.1510695000 20171114.

- **proctime**
  Returns user, system and real process times, in seconds since process start, as 3 double values. These are the number of seconds the process has spent actually running (user), doing system calls (system), and wall-clock time since process start (real). If the optional $arg argument is given, those values
are first subtracted. This allows a certain section of the process to be measured, instead of the entire cumulative process time (see example below). On error, values of -1 are returned.

- **procmem**  
  Returns virtual size and resident set size of the process, in bytes, as 2 `long` or `int64` values. On error, or if the current OS is unsupported, values of -1 are returned. Although the exact definition varies by platform, the virtual size is typically the total memory used by the process (code, data and stack), and the resident set size is the amount of that memory currently in physical RAM (as opposed to disk swap).

- **physmem**  
  Returns the amount of physical memory the system contains, in bytes, as a `long` or `int64` value. If this cannot be determined, -1 is returned. Added in version 3.01.981000000 20010131.

- **getppid or ppid**  
  Returns the parent process’ ID, as an integer. Not supported for Windows.

- **procexists**  
  For each process ID given in the $arg list, returns 1 if that process exists, 0 if not. Same as the procexists function (p. 333).

- **loadavg**  
  Returns the 3 system load averages as doubles, or -1 if unavailable or the OS is unsupported. Same as the loadavg function (p. 335).

- **cwd or pwd or getcwd**  
  Returns the current working directory of the process.

- **platform**  
  Returns a string describing the hardware, vendor, OS and file bit size that this executable was created for, and is probably running on. Returned as 4 values separated by dashes. Added in version 3.01.980200000 20010122. Note that this may vary from the actual platform that is running, e.g. the revision of the OS.

- **regquery $key $subkey $name [$default]**  
  Under Windows, queries the registry under the given $key and $subkey for the value associated with $name. $key must be one of the predefined `HKEY_...` values such as “HKEY_LOCAL_MACHINE”. $subkey is generally a string “path”. If there is no value defined, the $default value is returned if given. Under non-Windows platforms this function is unimplemented; the $default value is always returned. Added in version 3.01.976300000 20001208. regqueryvalue is an alias for regquery.

- **resourcestat $who $statName**  
  Returns the current process’s usage of the resource named by $statName, which is one of the following:

---

\[ int64 \] is only used when memory size could exceed the size of long on the system, and only in version 6.01 or later.
- UserTime
- SystemTime
- RealTime
- MaxResidentSetSize
- IntegralSharedMemSize
- IntegralUnsharedDataSize
- IntegralUnsharedStackSize
- MinorPageFaults
- MajorPageFaults
- Swaps
- BlockInputOps
- BlockOutputOps
- MessagesSent
- MessagesReceived
- SignalsReceived
- VoluntaryContextSwitches
- InvoluntaryContextSwitches

The optional $who parameter is one of self, children, both or thread; the default is self. This determines which resource group the value is returned from. Not all platforms support all resources, nor all $who values. Added in version 7.01.1384824000 20131118.

• umask
Returns the process's umask, as a chmod-style symbolic string. Added in version 4.00.997400000 20010809.

• getenv $var
Returns the value of environment variable named by $var. Note: Vortex variables are automatically initialized from environment variables; e.g. $PATH in Vortex contains the path from the environment. This call is less efficient, and should only be needed if environment variables have been changed since script start with syscp setenv. Added in version 4.00.997400000 20010809.

• availablespace $path
Returns the amount of free disk space in bytes (for non-root users) for each filesystem in $path, as an int64 value. If the free space cannot be determined, -1 is returned. On some old platforms an inaccurate value may be returned. Added in version 7.05.1450200000 20151215.

• freespace $path
Returns the amount of free disk space in bytes (for root) for each filesystem in $path, as an int64 value. If the free space cannot be determined, -1 is returned. On some old platforms an inaccurate value may be returned. Added in version 4.04.1080180000 20040324. In versions prior to 7.05.1450200000 20151215, this returned the same as availablespace, i.e. space for non-root users.
1.5. BUILTIN FUNCTIONS

- `totalspace $path`
  Returns the total disk space in bytes for each filesystem in $path, as an `int64` value. If the total space cannot be determined, -1 is returned. On some old platforms an inaccurate value may be returned. Added in version 7.05.1450200000 20151215.

- `usedspacepercent $path`
  Returns the used disk space for each filesystem in $path, as a `double` percent value. If the total space cannot be determined, -1 is returned. On some old platforms an inaccurate value may be returned. Added in version 7.05.1450200000 20151215.

- `hostname`
  Returns the machine’s hostname as configured. Added in version 4.04.1080000000 20040322.

- `fullhostname`
  Returns the machine’s hostname, fully-qualified via DNS. Added in version 4.04.1080000000 20040322.

- `user` or `realuser`
  Returns the real user name running the script. Added in version 4.04.1072150000 20031222. In version 7.00.1352762000 20121112 the alias `realuser` was added.

- `effectiveuser`
  Returns the effective user name running the script. Added in version 7.00.1352762000 20121112.

- `maxdescriptors`
  Returns the maximum number of open file descriptors possible, or -1 if unknown. Added in version 4.02.1037476167 20021116.

- `opendescriptors`
  Returns the current number of open file descriptors, or -1 if unknown. Added in version 4.02.1037476167 20021116.

- `processdescription`
  Returns process description (sans “texis: ” prefix) currently set by `<syscp processdescription>` (p. 321), or empty if none. Added in version 7.06.1522794000 20180403. Note that the returned description may be truncated from the value passed to `<syscp processdescription>`, due to limited original command line space in the process.

- `maxprocessdescriptionlen`
  Returns maximum length in characters of a process description that can be passed to `<syscp processdescription>` (p. 321) without truncation. Added in version 7.06.1522794000 20180403.

**DIAGNOSTICS**

`sysinfo` returns various OS-level parameters depending on its first argument.
EXAMPLE

To obtain the CPU utilization of an entire script, call `<sysinfo proctime>` at the end. In this example, we time a single function call, `<CPUhog>`, while ignoring any previous or later functions:

```bash
... ignore time of earlier calls here ...
<sysinfo proctime>
<start = $ret>
<CPUhog>
<sysinfo proctime $start>
CPUhog times: <LOOP $ret> $ret </LOOP>
...
```

CAVEATS

The `sysinfo` function was added in version 3.01.967500000 20000828.

Some options may be unavailable on some platforms, or may require additional user privileges (such as reading kernel memory). Vortex should not be run as `root` to obtain these privileges; use a lower privilege user that still has the required perms (e.g. `kmem`). No databases should be accessed when running as such a non-Texis user, to avoid making files inaccessible to other (Texis user) invocations of Vortex.

SEE ALSO

`syscp, sysutil`
1.5. BUILTIN FUNCTIONS

1.5.79  syscp – set system-specific information

SYNOPSIS

<syscp $what [$arg ...]>

DESCRIPTION

The syscp function sets various system- or OS-level parameters. The $what argument determines what is affected:

- **chdir $dir or cd $dir**
  Change the current working directory to $dir.

- **umask $mode**
  Change the process’s umask to $mode, which must be a chmod-style bitmask. These bits are ANDed (masked off) with the mode bits given during file open commands. Added in version 4.00.997400000 20010809.

- **setenv $varname $value [$overwrite]**
  Sets each environment variable named by $varname to the corresponding $value. If the environment variable already exists, it will be overwritten, unless the $overwrite argument (if present) is false (i.e. the default is to overwrite). Added in version 3.01.991350000 20010531. Notes:
    - setenv changes the true current process environment, not the Vortex variable(s) of the same name that may have been initialized at start.
    - Vortex variables can be assigned on the command line for a script, so it is not necessary to set the environment to pass variables to a sub-process <EXEC> script.
    - In version 5 and later, the ENVSET option to <EXEC> can be used to set environment vars for just the sub-process being <EXEC>ed, without modifying (with potential side effects) the current process’s environment.

- **unsetenv $varname**
  Clears environment variable $varname.

- **processdescription $desc**
  Sets process description to $desc (a string). The given value is prefixed with “texis: ” and copied to the live process command line, which is generally viewable with ps or Task Manager/Process Explorer. This lets a long-running Vortex script show its current state, task, progress, etc. in a limited but portable and easily visible way. If $desc is empty or unset, the description is cleared and the original command line restored. Setting or clearing a process description has no effect on $cmdlnargs, <vxinfo cmdlnargs>, nor

34While Task Manager and Process Explorer periodically refresh their displays ala Unix top, they may not refresh their copy of the process command lines as top does. Thus, monitoring changes in a Vortex script’s <syscp processdescription> with Task Manager or Process Explorer may require periodically restarting these viewers.
<vxinfo scriptargs>, Returns empty on success, “Truncated” silently if description was set but truncated (limited by original command line length), or “Failed” (and error message reported) on error. Added in version 7.06.1522794000 20180403. See also <sysinfo maxprocessdescriptionlen> (p. 319).

DIAGNOSTICS

syscp returns an empty string on success, otherwise a string describing the error.

EXAMPLE

<syscp chdir "/some/other/dir">

CAVEATS

The syscp function was added in version 3.01.975700000 20001201.

SEE ALSO

sysinfo, sysutil
1.5. **BUILTIN FUNCTIONS**

### 1.5.80 sysutil – file and system utilities

**SYNOPSIS**

```bash
<sysutil $action [option ...] $arg [...]]
```

**DESCRIPTION**

The `sysutil` function provides various file and system utilities. The `$action` argument determines the action; its possible values and the expected syntax of successive arguments are as follows:

- **`mkdir` or `md $dir [dir ...]`**
  Creates each directory named in `$dir` and successive arguments. If a permissions mode is given as an option (see below), the directory will be created with that mode; the default is 0777. In either case the mode is masked by the process's umask.

- **`chown` or `lchown $owner $file [file ...]`**
  Changes ownership of each file named by `$file` and successive arguments to the user and/or group named by the parallel value of `$owner`. If there are more file values than `$owner` values, the last `$owner` value is re-used. If the owner value has a period or colon appended, an optional group name may appear afterwards, in which case the group will be changed as well. If the group name is empty, the named user's default group will be used. If the named user is empty but a group is present (e.g. value starts with period or colon), only the group is changed; in this case the behavior is that of `chgrp` or `lchgrp`. If `lchown` is used, the owner of symlinks is changed instead of the file the symbolic link points to. **Note:** These actions are unimplemented under Windows.

- **`chgrp` or `lchgrp $group $file [file ...]`**
  Changes the group of each file named by `$file` and successive arguments to the group named by the corresponding value of `$group`. If there are more file values than `$group` values, the last `$group` value is re-used. If `lchgrp` is used instead of `chgrp`, the group of symlinks is changed instead of the file the symbolic link points to. **Note:** These actions are unimplemented under Windows.

- **`chmod $mode $file [file ...]`**
  Changes the file permissions of each file named by `$file` and successive arguments to the mode named by the parallel value of `$mode`. If there are more file values than `$mode` values, the last `$mode` value is re-used. A mode value may be octal numeric (e.g. "0666") or symbolic ("ugo+rw").

- **`attrib $attrs $file [file ...]`**
  Changes the file attributes of each file named by `$file` and successive arguments to the value named by the parallel value of `$attrs`. If there are more file values than `$attrs` values, the last `$attrs` value is re-used. An attribute value may be octal numeric (e.g. "040" for the archive bit) or symbolic. Symbolic values follow a similar syntax to `chmod` symbolic values:

  ```plain
  [+|-|=]token[,token...][,+|=]token[,token...]
  ```

  i.e. one of the operator characters plus, minus or equals, followed by zero or more comma-separated attribute tokens. The possible tokens are: `readonly`, `hidden`, `system`, `volumelabel`, `directory`, `archive`, `readwrite`, `hidden`...
device, normal, temporary, sparsefile, reparsepoint, compressed, offline, notcontentindexed, encrypted. Not all attributes are settable, e.g. the directory and device attributes cannot be modified. The normal attribute, if set, must be set alone and with the set (=) operator. This is a Windows-specific operation; on other OSes (e.g. Unix), attrib may be emulated to a limited extent, and chmod is preferred. Added in version 5.01.1245200000 20090616.

- **rm** or **del** or **delete** $file [$file ...]
  Removes each file named by $file and successive arguments.

- **rmdir** or **rd** $dir [$dir ...]
  Removes each directory named by $dir and successive arguments.

- **mv** or **rename** $src [$src ...] $dest
  Moves each file named by $src and successive arguments (except the last) to the corresponding value of $dest. If there are more source values than $dest values (either as given or due to wildcard globbing or recursion), the $dest value must exist and be a directory, in which case the source files are moved into it. Moving files across filesystems is supported.

- **cp** or **copy** $src [$src ...] $dest
  Copies each file named by $src and successive arguments (except the last) to the corresponding value of $dest. If there are more source values than $dest values (either as given or due to wildcard globbing or recursion), the $dest value must exist and be a directory, in which case the source files are copied into it.

- **link** or **symlink** $target [$target ...] $name
  Creates a link $name to the specified $target. If there are more targets than names, the last $name must be a directory, in which case the links are created in it. If symlink is used instead of link, a symbolic link instead of a hard link is created. **Note:** These actions are unimplemented under Windows.

- **sync**
  Calls sync() to flush system write buffers to disk. Not implemented under Windows. Added in version 5.01.1099669738 20041105.

- **touch** $file [$file ...]
  Touches – i.e. updates last-access/last-modification times to “now” – each $file, creating it if nonexistent. Can be passed nocreate, date, reference, setaccess, setmodify options (see below). Added in version 6.00.1302850000 20110415.

After the $action argument and before any successive arguments, the following options may be given:

- **msg**
  Turns on the generation of file-related putmsg calls for this action. Overrides the current filemsg value set with vxcp (p. 284).

- **nomsg**
  Turns off the generation of file-related putmsg calls for this action. Overrides the current filemsg value set with vxcp (p. 284).
1.5. BUILTIN FUNCTIONS

- **glob**
  Enables globbing of source filenames: the characters “*” and “?” become significant as wildcard characters. Note that all filename(s) expanded from a given source argument still correspond to the same destination argument (if applicable for the action).

- **recurse**
  Enables recursive descent of source directories, e.g. for removing or copying an entire directory tree. For `cp`, the additional descended path will be appended to the destination path, so that the copied tree has the same hierarchy as the source.

- **force**
  Force the removal of files or directories, even if the file does not have write permission.

- **preserve**
  Preserve the file attributes (owner, permissions) of copied files, for `cp`. (This flag is always on for `mv`.)

- **asis**
  Copy non-directories “as-is” for `cp`, e.g. symlinks are created for symlinks. The default is to copy non-directories as regular files. Not applicable under Windows. (This flag is always on for `mv`.)

- **parents**
  For `mkdir`, create the parents of the named directory if they do not already exist. Useful for creating a multi-level directory path in one call.

- **nocreate**
  For `touch`, do not create the file if it does not exist. Default is to create a new file if nonexistent.

- **date $date**
  For `touch`, next arg is a TeXis-parseable date to use for access/modification times, instead of current time.

- **reference $file**
  For `touch`, next arg is a file to copy access/modification times from, instead of using current time.

- **setaccess**
  For `touch`, set access time. Default if neither `setaccess` nor `setmodify` is given is to set both times.

- **setmodify**
  For `touch`, set modify time. Default if neither `setaccess` nor `setmodify` is given is to set both times.

- **nnn or [ugoa]=[rwx]**
  For `mkdir`, an optional file permission mode may be given to use instead of the default 0777.

- **--**
  Signifies the end of options. This is useful to avoid misinterpreting following file arguments as options, if they may have the same name.
DIAGNOSTICS

sysutil returns an empty string on success, otherwise a string describing the first error encountered.

EXAMPLE

<sysutil rm recurse force $dir>

The above example deletes the entire directory tree rooted at $dir.

CAVEATS

The sysutil function was added in version 3.01.984600000 20010314.

SEE ALSO

stat, sysinfo, syscp
1.5. BUILTIN FUNCTIONS

1.5.81 stat – obtain file size and attributes

SYNOPSIS

<stat [options] $path[ ]/> [</stat>]

DESCRIPTION

The stat function obtains information about file(s). It combines functionality from the Unix utilities ls, stat() and find. For each file named in the $path argument (plus others, depending on options below), the following variables are set in parallel:

- $ret (string)
  The file path. This is usually a value from the $path argument, but if GLOB or MAXDEPTH is set, new paths may be returned.

- $ret.err (string)
  The error from stat() for this file, or empty if no error. Thus, if the file named by $path does not exist, $ret.err will be non-empty. Note that it is possible for errors to occur on files not specified in the original $path list, if GLOB or MAXDEPTH is set. If $ret.err is non-empty, any variables derived from a stat() call (e.g. $ret.size) will also be unset, empty or 0.

- $ret.depth (long)
  The number of directories traversed from an original $path argument to this file, excluding any initial filename globbing. Top-level paths will thus have a depth of 0.

- $ret.symlink (string)
  The raw target of the symlink. For non-symlink files and platforms that do not support symbolic links, this is empty.

- $ret.sympath (string)
  The target of a symlink, as a corrected path from the top-level $path argument. For non-symlink files and platforms that do not support symbolic links, this is empty.

- $ret.size (long, int64 or double)
  The size of the file, in bytes. On systems where file sizes can exceed the size of a long (e.g. Solaris 2.6/2.7/2.8), the value is returned as an int64 (version 5 and earlier returned double).

- $ret.owner (string)
  The owner of the file.

- $ret.group (string)
  The group of the file.

- $ret.isrd (long)
  1 if the file is readable with current permissions, 0 if not.
- $\text{ret.iswr}(\text{long})$
  1 if the file is writable with current permissions, 0 if not.

- $\text{ret.isex}(\text{long})$
  1 if the file is executable with current permissions, 0 if not.

- $\text{ret.mode}(\text{string})$
  Type and permissions of the file, as a 10-character symbolic string ala Unix `ls`. The first character denotes the type of file: “d” for a directory, “-” for a regular file, “b” for a block device, “c” for a character device, “p” for a FIFO or pipe, “l” for a symlink, “s” for a socket. The next 3 characters are “r”, “w” and “x” respectively, to indicate read, write and execute permission for the file owner, or “-” to indicate the permission is not given. Under Unix, the next 3 characters are the same, for the group. The last three are the same, for others. The user execute bit may be “s” if the set-uid bit is also set, or “S” if the set-uid bit is set without execute. The group execute bit may be “s” if the set-gid bit is also set, or “S” if the set-gid bit is set without execute. The other execute bit may be “t” if the save-text (sticky) bit is also set, or “T” if the bit is set without execute.

- $\text{ret.attrib}(\text{string})$
  List of file attributes of the file, as a comma-separated list of zero or more of the following tokens: `readonly`, `hidden`, `system`, `volumelabel`, `directory`, `archive`, `device`, `normal`, `temporary`, `sparsefile`, `reparsepoint`, `compressed`, `offline`, `notcontentindexed`, `encrypted`. This is a Windows-specific return value: on other OSes, $\text{ret.attrib}$ may be emulated to a limited extent (e.g. under Unix `readonly` is set if the file is not writable), and $\text{ret.mode}$ contains more details. Added in version 5.01.1245200000 20090616.

- $\text{ret.atime}(\text{date})$
  The last-access time of the file, which is generally the last time a process read from the file.

- $\text{ret.mtime}(\text{date})$
  The last-modify time of the file.

- $\text{ret.ctime}(\text{date})$
  The last-change time of the file, i.e. the last time its attributes were changed.

- $\text{ret.nlinks}(\text{long})$
  The number of hard links to the files (if the filesystem/platform supports it).

- $\text{ret.devtype}(\text{long})$
  The device type the file is on (if the filesystem/platform supports it).

- $\text{ret.dev}(\text{long})$
  Under Unix, the device major and minor number (combined). Under Windows, this usually indicates what drive the file is on: 0 for A:, 1 for B:, etc. (For UNC paths this may be the drive that the process is on, not the file; this is apparently a limitation of the Windows `stat()` implementation.)

- $\text{ret.ino}(\text{long})$
  The inode of the file (if the filesystem/platform supports it).

- $\text{ret.blks}(\text{long})$
  The number of blocks consumed by the file, if the filesystem/platform supports it.
The preferred block size for file transfers on the device, if the filesystem/platform supports it.

The ending \texttt{/stat} tag is optional: if present, the function becomes a looping block statement, with the return variables looped over and any statements inside the block executed for each iteration; \texttt{$\text{loop}$} and \texttt{$\text{next}$} are also set then. \texttt{\textless \textbackslash \text{break} \textgreater} (p. 53) may be used to exit the loop.

The following options may be set before the \texttt{$\text{path}$} argument:

- \texttt{ROW}
  Do not accumulate return variables into arrays, and do not enter a loop context for them; each iteration’s values will replace the previous. This option is recommended if the return values are not needed in future iterations; it saves memory.

- \texttt{SKIP=n}
  Skip the first \texttt{n} return values. For example, to list just the contents of a single directory, \texttt{SKIP=1} with \texttt{MAXDEPTH=1} will skip the initial value (the directory itself).

- \texttt{MAX=n}
  Return at most \texttt{n} values, after \texttt{SKIP}, globbing and recursion if any.

- \texttt{MAXDEPTH=n}
  Descend at most \texttt{n} directories deep from the top-level \texttt{$\text{path}$} argument values (after globbing). The default is 0, i.e. do not descend directories. A negative value indicates no limit.

- \texttt{NAME= wildcard}
  Only return files whose name (not path) matches the \texttt{wildcard} specification. Similar to the \texttt{-name} option to the Unix \texttt{find} utility.

- \texttt{ALL}
  Return all entries from a traversed directory; do not skip \texttt{“.”} and \texttt{“..”}.

- \texttt{SAMEDEV}
  Stay on the same device as the current \texttt{$\text{path}$} argument; do not cross filesystems when traversing directories if \texttt{MAXDEPTH} specified. Same as the \texttt{-xdev} or \texttt{-mount} option to the Unix \texttt{find} utility. May not work correctly under Windows.

- \texttt{DEPTHFIRST}
  When traversing directories (if \texttt{MAXDEPTH} set), return a directory’s contents before the directory name itself. The default is to return the directory name before returning its contents.

- \texttt{SYMLINK}
  Return information about symlinks, not the files they point to; i.e. use \texttt{lstat()} instead of \texttt{stat()}. This does not affect the value of \texttt{$\text{ret.symlink}$} or \texttt{$\text{ret.sympath}$}, nor the traversal of directories (see \texttt{FOLLOWSYM} option); it will affect other \texttt{stat()}-dependent variables such as \texttt{$\text{ret.mode}$} however.

- \texttt{FOLLOWSYM}
  Follow symbolic links that point to directories when traversing for \texttt{MAXDEPTH}; the default is not to. \textbf{Note:} this can cause the same directory tree to be traversed many times, e.g. if a symbolic link points to an upper-level directory.
CHAPTER 1. TEXIS WEB SCRIPT

- **GLOB**
  Do shell-style file globbing: expand wildcards ("*" and "?") in the original $path argument's values, and return values for the resulting paths. The expanded values for a given $path wildcard are sorted ascending by name. Without this flag, wildcard characters have no special meaning and are interpreted literally.

- **SORT=method**
  Sort the contents of each directory traversed (via MAXDEPTH) by method, which defaults to “name”. The possible values are “none”, “name”, “size”, “atime”, “mtime” or “ctime”. Note that each traversed directory is sorted individually, not the entire result set.

- **ASC**
  Sort each descended directory’s files in ascending order (the default).

- **DESC**
  Sort each descended directory’s files in descending order.

**DIAGNOSTICS**

`stat` returns the file path in $ret, plus various other $ret... variables as listed above. If the looping syntax is used (a closing `<stat>` tag), $loop and $next are set as well as the other variables.

**EXAMPLE**

```xml
<stat /very/important/file>
<IF $ret.err neq "">  The file does not exist! ($ret.err)
</IF>
...

Contents of directory $dir:

<stat ROW MAXDEPTH=1 SKIP=1 ALL $dir>
  <fmt "%s %8s %8s %10kd %at %s\n"
    $ret.mode $ret.owner $ret.group
    $ret.size "%b %d %Y" $ret.mtime $ret>
</stat>
```

The top part of this example checks for the existence of a file, and reports if it cannot be found. The bottom part of the example emulates an `ls` or `dir` of the directory $dir, and prints out some information on the files contained therein: setting MAXDEPTH to 1 ensures the directory contents are returned as well, SKIP=1 skips the directory name itself, and ALL ensures that “.” and “..” are returned too.
CAVEATS

The `stat` function was added in version 3.01.982000000 20010212.

Certain return variables are platform-dependent, such as `$ret.dev`, `$ret.symlink` and `$ret.sympath`.

Using the `FOLLOWSYM` option can cause repetitive, copious and useless return values if symbolic links point to directories, as the resulting filesystem loop will be followed.

SEE ALSO

`sysutil`, `read`, `WRITE`
1.5.82 getpid – get process id

SYNOPSIS

<getpid>

DESCRIPTION

The getpid function returns the process id (pid) of the Vortex process running the script. This may be used to identify a process among multiple simultaneous Vortex invocations.

DIAGNOSTICS

The getpid function returns the integer Vortex process ID.

CAVEATS

The getpid function was added in version 2.1.900900000 19980720.

Although a process id is unique among currently-running processes, it may be re-used eventually once a process exits.

SEE ALSO

sysinfo, procexists, kill
1.5. BUILTIN FUNCTIONS

1.5.83 procexists – see if process exists

SYNOPSIS

<procexists $pid>

DESCRIPTION

For each process id in $pid, the procexists function returns 1 if the process exists, 0 if not.

DIAGNOSTICS

The procexists function returns 1 if the process exists, or 0 if not, for each $pid.

CAVEATS

The procexists function was added in version 2.1.900900000 19980720.

Although a process id is unique among currently-running processes, it may be re-used eventually once a process exits.

SEE ALSO

sysinfo, getpid, kill
1.5.84  kill – terminate or send a signal to a process

SYNOPSIS

<kil l $pid [$sig]>

DESCRIPTION

To each process id in $pid, the kill function sends the signal in the corresponding value of $sig. If fewer values of $sig are present than $pid, the last is re-used; if none are given, SIGTERM(15) is used. Signals may be given as numbers or names (e.g. “SIGHUP” and 1 are generally equivalent).

It is generally advisable to use the default or SIGTERM signal first when killing a process, so that it has a chance to clean up and potentially avoid corruption issues. Only after SIGTERM fails and several seconds have passed should a hard SIGKILL be used.

DIAGNOSTICS

The kill function returns 1 if the call succeeds, 0 if not. In version 3.01.983500000 20010301 and earlier, nothing is returned.

CAVEATS

The kill function was added in version 2.1.900900000 19980720.

For Windows, version 5.01.1171938352 20070219 and later map signal 0 to procexists, SIGINT/SIGBREAK to a Ctrl-Break event, and SIGTERM to a Texit Terminate event (soft kill, works with most Texit processes only). All other signals are mapped to TerminateProcess, which is a “hard” kill of a process and should only be used as a last resort. Prior to Windows version 5.01.1171938352 20070219, all signals mapped to TerminateProcess. Prior to Windows version 3.01.983500000 20010301, the kill function had no effect. Use the procexists function instead of <kill $pid 0> to test for the existence of a process.

SEE ALSO

 getpid, procexists
1.5.85 loadavg – return system load averages

SYNOPSIS

<loadavg>

DESCRIPTION

The loadavg function returns a list of 3 double values for the system load averages, generally over the last 1, 5 and 15 minutes respectively.

DIAGNOSTICS

loadavg returns a list of 3 double values representing the system load averages. In the event of an error, -1 is returned for one or more of the values.

EXAMPLE

<loadavg>
<IF $ret gt 3.0> Sorry; please try again later. <exit> </IF>

CAVEATS

The loadavg function was added in version 2.1.894000000 19980501. Solaris support was added in version 2.6.928300000 19990602, and Irix support in version 2.6.929000000 19990610.

On some platforms loadavg is unimplemented. On others it will not function if the process cannot read kernel memory (it may have to be in group kmem). In such cases -1 is returned to indicate failure. Running the script via the Texis Web Server (vhttpd) may help, since vhttpd is usually started as root.

The time periods for each load average number, and the definition of “load average” itself, vary by platform. On some platforms loadavg can use noticeable system time when run in a high load environment. Running the script via the Texis Web Server can reduce this load, by pre-initializing some system data.

SEE ALSO

sysinfo
CHAPTER 1. TEXIS WEB SCRIPT

1.6 XML API

1.6.1 Overview

The Vortex XML API is a set of SQL functions that provide a robust method for interacting with XML files and XML data in Vortex. The Vortex XML API provides functionality for:

- Reading XML data from a file, and writing XML data to a file.
- Reading XML data from a string in memory, and serializing data to a string.
- Modifying XML data in memory, including adding, removing, moving, and changing elements/attributes.
- Combining multiple XML documents into a single document, and splitting a document into multiple documents.
- Navigating XML data quickly and easily via XPath 1.0.
- Transforming XML data with XSL stylesheets.

The XML API is also fully namespace aware, which includes reading and modifying namespace prefixes/URIs, changing namespace assignments, etc.

1.6.2 Character Encodings

There are two different items to discuss related to encodings; reading/writing XML, and working internally.

The XML API can read and write many character encodings, leveraging the power of the GNU libiconv library. The reading and writing encodings need not be similar. For example, a SHIFT_JIS document can be written as UTF-8, and vice-versa.

When working within the library, everything is UTF-8 regardless of what character encoding it was read from or will be written to. This deserves stressing:

- The XML API can parse many character encodings.
- All data extracted through the XML API is UTF-8.
- All data changed or inserted through the XML API must be UTF-8.
- The XML API can serialize documents in many character encodings.

This means that a document may exist on disk in ISO-8859-1, but when the XML API parses it and you call xmlTreeGetContent() to get the text from an element, you’ll get UTF-8 data. If the file exists on disk in ASCII, calling xmlTreeGetContent() will still give UTF-8 data.
1.6. **XML API**

Similarly, regardless of whether a document will be outputted in BIG5, ISO-8859-7, UTF-32, etc., when adding a new element with `xmlTreeNewElement()`, the name and contents must be given in UTF-8.

This may sound restricting, but it’s actually liberating in that when working in code, you never have to worry about what encoding the file was read from, or what encoding it will be written out as. Always use UTF-8.

The default encoding when working in Vortex is already UTF-8 (unless manually changed with `<urlcp charsettxt>`). If you have data that you need to convert to UTF-8, you can use the `<urlutil charsetconv>` Vortex function.

### 1.6.3 Tree vs. Streaming APIs

There are two methods for interacting with XML in Vortex: a tree API, and a streaming API. Which should you use? That all depends on what kind of features you need, and what memory/speed requirements you may have.

**The Tree API**

The tree API, `xmlTree`, reads and parses an entire XML document into a hierarchical tree when it is opened and keeps it in memory. This provides a tree that is available for random access to the entire XML document, allowing things like traversing in both directions, and concurrent reading, modifying, and creating data.

**The Streaming API**

The streaming API does not hold the entire document in memory. It is actually composed of two separate interfaces, `xmlReader` and `xmlWriter`. These operate more closely to normal file handles. When using `xmlReader`, you are given a reader object that you advance linearly through the file, examining the current element. Similarly, with the `xmlWriter`, you give it data, serially, that is written to the file or string.

### Comparing Features

Streaming API:

- The streaming API is limited in that it can read, or it can write. The `xmlReader` and `xmlWriter` APIs are separate, so there’s no modifying XML that is being read with `xmlReader`, and no looking back at data that’s come out of `xmlWriter`.

- The streaming APIs are also linear - the `xmlReader` can only examine the current element, meaning operations such as “get the current element’s parent” are impossible. Similarly, the `xmlWriter` cannot change data that has already been send to the API for writing.

- XPath processing is also unavailable for the `xmlReader` interface, since previous and future elements are not kept in memory to be analyzed.
• XSL Processing is unavailable for the xmlReader interface, as XSL processing relies on random access and XPath expressions during the transformation.

Tree API:

• The tree API does not have these limitations. Because the entire XML document is held in memory, data can be read and written simultaneously.
• The tree can be accessed sequentially, randomly, or any desired method. A tree could be constructed from the inside out, if so desired.
• XPath processing is available for the tree API, allowing for a very easy, very powerful method of traversal for XML data.
• XSL Processing is available for the tree API, allowing the application of a stylesheet to produce XML or HTML data.

Memory Usage

Streaming API:

• Both the xmlReader and xmlWriter have a very small memory footprint, only enough for the state information of the XML parser, and in the case of xmlReader, the current element. The memory consumed stays relatively constant, regardless of the size of the XML document being dealt with.

Tree API:

• The xmlTree interface holds the entire tree in memory, and requires around 4 times the size of the XML document. For example, the XML-1.0 recommendation is about 150KB and, when parsed, uses around 650KB of memory.

Parsing Speed

Streaming API:

• Both xmlReader and xmlWriter are very fast, given that they are little more than wrappers for direct reading/writing that process extra information, or add extra information, respectively. The amount of time to open a handle XML data is constant, as it is not affected by the size of the XML data.

Tree API:

• With xmlTree, the entire XML document needs to be read and parsed before any operations can be performed on it, so it is affected by the size of the XML data. The xmlTree API is very fast for an XML parser, but there will still be a noticeable delay when parsing 900MB of XML data, for example.
1.6. XML API

Which to Use

So which method do you use? If you have strict memory requirements (or are working with extremely large files), then you may be forced to use the Streaming APIs. Otherwise it’s your choice between the feature-full Tree API or the smaller, quicker Streaming API.

1.6.4 Data Types

The XML APIs introduce some new internal data types to Vortex. Note that these are not SQL data types, and cannot be stored outside of memory, e.g. in a table.

These types cannot be printed directly. For example just printing $var when it contains an xmlNode will simply print (xmlNode). The methods for converting them to strings are provided in their individual descriptions.

As with all Vortex data, there is no direct memory management necessary. All allocating and freeing is handled by Vortex for you. However, to conserve memory, it is recommended to make sure the variable(s) used are either deleted (i.e. <$doc = >) or go out of scope (i.e. use <local> variables) when done, especially when accessing large documents with the tree API.

xmlNode

The xmlNode is the general object type in the xmlTree API. With few exceptions, most things worked with in the xmlTree API are xmlNode’s. They are used to represent elements, text objects, attributes, entities, comments, processing instructions, etc.

In general you don’t “print” xmlNode’s directly, they’re part of a larger XML document that gets formatted via the xmlDoc root object.

xmlDoc

The xmlDoc is a special case of xmlNode. Each XML document has a single xmlDoc that is that the root of the document. It can be used in many of the same functions as xmlNode’s (such as xmlTreeGetChildren()), but also contains document-wide information, such as character encoding.

xmlDoc’s can be saved to a file and converted to strings with the xmlTreeSaveDoc() (p. 468) and xmlTreePrintDoc() (p. 464) functions, respectively.

Also see the xmlDoc vs. Root Element section (p. 350) for details on the differences between xmlDoc’s and the root element of a document.

xmlNs

The xmlNs structure is how namespaces are stored within the xmlTree. While the xmlNode (which represents attributes and elements) has name and contents, xmlNs has a prefix and a URI.
These structures are also much less interlinked than xmlNode’s, so functions like xmlTreeGetPrevious() and xmlTreeGetParent() are not available. Only xmlTreeGetNext() can be used with xmlNs structures, which gives the next namespace declared on the element.

See the Using Namespaces section (p. 352) below for more information on namespaces and xmlNs objects.

xmlXPath

The xmlTree API supports the XPath 1.0 W3C Recommendation. When you want to perform XPath searches on XML data, you create a xmlXPath object from the xmlDoc. Then namespaces can be registered to the XPath, and searches can be performed through it.

See the Using XPath section (p. 347) for more information.

xsltStylesheet

When you want to apply an XSL stylesheet to an XML document, you create a xsltStylesheet object from the stylesheet you want to use (either a file, a string, or a doc). That xsltStylesheet can then be applied to any number of XML documents.

See the Using XSL section (p. 348) for more information.

xmlReader

When working with the xmlReader interface, there’s only one object you use, which is the xmlReader itself. It is used similarly to a file handle - you give it a data source (either a file or a string) and read data from it.

See the Using xmlReader section (p. 354) for more information.

xmlWriter

When working with the xmlWriter interface, there’s only one object you use, which is the xmlWriter itself. It is used similarly to how you would use a writing file handle - you simply give it data you want converted into XML.

See the Using the xmlWriter section (p. 356) for more information.

1.6.5 Using the xmlTree

This section discusses some of the details of the structure of the trees created when parsing an XML document with the xmlTree interface and how to use them.
The samples/xmlapi directory contains the following code examples:

- xmlTree01_New - shows many ways of creating new nodes in an XML tree
- xmlTree02_Get - shows ways of extracting information from an XML document
- xmlTree03_Sibling - shows many of the tree traversal mechanisms
- xmlTree04_Ns - shows how to resolve and examine namespace information in a tree
- xmlTree05_SetNs - shows how to create and set namespace information in a tree
- xmlTree06_Copy - shows how to make copies of nodes in an XML tree
- xmlTree07_Encoding - shows how to use various input/output encodings
- xmlTree08_XPath - shows how to use XPath expressions
- xmlTree09_DTD - shows how to extract DTD information
- xmlTree10_XSL - shows how to apply an XSL stylesheet

Below is a small example program that shows how the xmlTree API can be used to open an XML document, add content to an element, add another element, and print the output.

```
<capture>
<Result>
  <Url>http://www.example.com/page.htm</Url>
  <ResultTitle>Generic Page Title</ResultTitle>
</Result>
</capture>

$xmlRaw=$ret
$doc = (xmlTreeNewDocFromString($xmlRaw, 'XML_PARSE_NOBLANKS'))
$root = (xmlTreeGetRootElement($doc))
$title = (xmlTreeGetChildren($root, 'ResultTitle'))
$titleText = (xmlTreeGetContent($title))
Existing title is "$titleText"
$ret = (xmlTreeAddContent($title, ", now with 12% more fun"))
$ret = (xmlTreeNewElement($root, 'Abstract', 'content content'))
$output = (xmlTreePrintDoc($doc, 'INDENT'))
after changes doc is:
$output
```

Opening / Creating a Document

All work with the xmlTree API starts with the document, whether you’re using existing data or starting from scratch. There are three ways to obtain an xmlDoc object:

- Reading an XML document from a file (xmlTreeNewDocFromFile(), p. 454)
• Reading an XML document from a string (xmlTreeNewDocFromstring(), p. 456)
• Starting a new document (xmlTreeNewDoc(), p. 453)

EXAMPLE

```xml
<$rawXml = "<simpleExample>read me</simpleExample>">
<$doc = (xmlTreeNewDocFromstring($xmlRaw))>
<$root = (xmlTreeGetRootElement($doc))>
...
<$doc = (xmlTreeNewDocFromFile('sample.xml'))>
<$xpath = (xmlTreeNewXPath($doc))>
...
<$doc = (xmlTreeNewDoc('1.0'))>
<$root = (xmlTreeNewElement($doc, 'rootNode'))>
```

SEE ALSO
xmlTreeNewDoc, xmlTreeNewDocFromFile, xmlTreeNewDocFromString

Saving

xmlDoc’s can be written to a file with xmlTreeSaveDoc(), or converted to text with xmlTreePrintDoc().

EXAMPLE

```xml
<$ret = (xmlTreeSaveDoc($doc, 'output.xml'))>
<$output = (xmlTreePrintDoc($doc))>
output is $output
```

SEE ALSO
xmlTreePrintDoc, xmlTreeSaveDoc

Traversing the Tree

XML trees are very inter-linked documents. There are functions for getting children (like xmlTreeGetChildren()), getting various siblings (xmlTreeGetNext()), and the parent (xmlTreeGetParent()).
Attributes exist outside of the normal hierarchy. You can get all (or specific) attributes with xmlTreeGetAttributes(), or you can begin iterating over them with xmlTreeGetFirstAttribute().

Namespaces also exist outside of the normal hierarchy. See the Using Namespaces (p. 352) section for more information.

EXAMPLE

```xml
<capture>
<rootNode><name>John Doe</name><age>34</age></rootNode>
</capture>
$xmlRaw = $ret
$doc = (xmlTreeNewDocFromString($xmlRaw))
$root = (xmlTreeGetRootElement($doc))
$name = (xmlTreeGetChildren($root, 'name'))
$nameText = (xmlTreeGetContent($name))
$age = (xmlTreeGetNext($name))
$sageText = (xmlTreeGetContent($age))
name is $nameText, age is $ageText
```

SEE ALSO

xmlTreeGetDoc, xmlTreeGetRootElement, xmlTreeGetChildren, xmlTreeGetFirstChild, xmlTreeGetNext, xmlTreeGetPrevious, xmlTreeGetParent, xmlTreeGetFirstAttribute, xmlTreeGetAttributes, xmlTreeGetNs, xmlTreeGetNsDef, xmlTreeLookupNsPrefix, xmlTreeLookupNsURI

Creating Nodes

New nodes are added to an xmlTree by various new functions. They all take a parent, which specifies where the new node should be created, and either a name and optional content (in the case of elements and attributes), or just content (comments, CDATA, etc).

EXAMPLE

```xml
$doc = (xmlTreeNewDoc( '1.0'))
$root = (xmlTreeNewElement($doc, 'rootNode'))
$name = (xmlTreeNewElement($root, 'name', 'John Doe'))
$age = (xmlTreeNewElement($root, 'age', '23'))
$ret = (xmlTreeNewAttribute($age, 'type', 'years'))
$output = (xmlTreePrintDoc($doc, 'INDENT'))
output is $output
```
SEE ALSO
xmlTreeNewElement, xmlTreeNewPI, xmlTreeNewText, xmlTreeNewCDATA, xmlTreeNewComment, xmlTreeNewAttribute, xmlTreeNewNs

Getting Information

There are functions for getting information about a node, such as name and content. XmlNode’s all use the same functions, whether they’re elements, attributes, comments, etc.

XmlNode objects are different in that they have a prefix and URI instead of name and content. They have their own functions to reflect this.

EXAMPLE

```
<capture>
<rootNode><name>John Doe</name><age>34</age></rootNode>
</capture>$xmlRaw = $ret
$$doc = (xmlTreeNewDocFromString( $xmlRaw))>
$$root = (xmlTreeGetRootElement($doc))>
$$name = (xmlTreeGetName($root))>
$$type = (xmlTreeGetType($root))>
$$nameText = (xmlTreeGetChildrenContent($root, ’name’))>
The root $name’s type is $type, name’s text is $nameText
```

SEE ALSO
xmlTreeDumpNode, xmlTreeGetName, xmlTreeGetChildrenContent, xmlTreeGetContent, xmlTreeGetAllContent, xmlTreeGetLine, xmlTreeGetType, xmlTreeIsBlankNode, xmlTreeGetAttributeContent, xmlTreeGetNsURI, xmlTreeGetNsPrefix

Setting Information

There are relatively fewer functions for setting data, reflecting the fact that the only real data stored in the nodes are name and content.

Once again XmlNode get their own versions for the prefix and URI.

EXAMPLE
1.6. XML API

SEE ALSO
xmlTreeSetName, xmlTreeSetContent, xmlTreeAddContent, xmlTreeSetNsPrefix, xmlTreeSetNsURI

Moving Objects

All of the moving functions remove a node from its original location and add it to another location among other nodes. The existing nodes are not affected, with the exception of xmlTreeSetRootElement(), as there can only be one root element.

EXAMPLE

SEE ALSO
xmlTreeAddChild, xmlTreeAddChildList, xmlTreeAddNextSibling, xmlTreeAddPrevSibling, xmlTreeAddSibling, xmlTreeSetRootElement, xmlTreeSetNs
Deleting Objects

Objects aren’t actually deleted in the xmlTree API, they’re unlinked. This removes all references to the node from its tree, which removes it from the output. The XML API takes care of the deletion when necessary.

The object is not deleted until it is no longer accessible through any Vortex variable, so an unlinked node is still fully accessible from functions like xmlTreeGetName() and xmlTreeGetContent(), although things like xmlTreeGetParent() will not behave as before the unlinking.

EXAMPLE

```xml
<capture>
<rootNode><name>John Doe</name><age>34</age></rootNode>
</capture><$xmlRaw = $ret>
<$doc = (xmlTreeNewDocFromString( $xmlRaw, 'XML_PARSE_NOBLANKS'))>
<$root = (xmlTreeGetRootElement($doc))>
<$age = (xmlTreeGetChildren($root, 'age'))>
<$ret = (xmlTreeUnlinkNode($age))>
<$output = (xmlTreePrintDoc($doc, 'INDENT'))>
output is $output

SEE ALSO
xmlTreeUnlinkNode

Copying Objects

The copying functions behave like the moving functions, except they leave the original copy of the node intact.

There are no namespace copying functions, as it’s rare to need to actually copy a namespace. It’s more common to simply use xmlTreeSetNs() to tell an object to use a namespace. See the Using Namespaces (p. 352) section for more information.

EXAMPLE
<capture>
<rootNode><name>John Doe</name><age>34</age></rootNode>
</capture>
$xmlRaw = $ret
$doc = (xmlTreeNewDocFromstring( $xmlRaw, 'XML_PARSE_NOBLANKS'))
$root = (xmlTreeGetRootElement($doc))
$age = (xmlTreeGetChildren($root, 'age'))
$age2 = (xmlTreeCopyNode($root, $age, 1))
$output = (xmlTreePrintDoc($doc, 'INDENT'))
output is $output

SEE ALSO
xmlTreeCopyDoc, xmlTreeCopyNode, xmlTreeCopyNodeList, xmlTreeCopyAttribute, xmlTreeCopyAttributeList

Using XPath

XPath expressions provide a powerful way of addressing parts of an XML document. The W3C XPath recommendation can be found at http://www.w3.org/TR/xpath. Many good XPath tutorials can be found on the web, such as http://www.w3schools.com/xpath/.

Essentially, XPath expressions allow you to search a document for nodes that match a given criteria. Note that it finds nodes: this means it can locate not only elements, but attributes, text content, etc.

To use XPath in the xmlTree API, a XPath object is first created with xmlTreeNewXPath(). If any namespaces are needed in the expression, they must be registered with the xpath object via xmlTreeXPathRegisterNs(). XPath expressions can then be executed with xmlTreeXPathExecute()
CHAPTER 1. TEXIS WEB SCRIPT

If you have XML data stored in the field of a database, you can use xmlTreeQuickXPath() to easily extract data from it for comparison:

```
select name from books
where xmlTreeQuickXPath(data, '/rootNode/item') = 'Bill'
```

See the sample xmlTree08_XPath for more and larger examples using XPath and QuickXPath.

SEE ALSO

xmlTreeNewXPath, xmlTreeXPathExecute, xmlTreeXPathRegisterNs, xmlTreeQuickXPath

Using XSL

XSL Transformations are performed in the xmlTree API by applying xs1tStylesheet objects to xmlDoc documents.

xs1tStylesheet objects are acquired by reading an XSL stylesheet from a file, from a string, or from an XSL document that's already been parsed as an xmlDoc (using xs1tParseStylesheetFile, xs1tParseStylesheetString, or xs1tParseStylesheetDoc, respectively).

The xmlDoc's that the stylesheets are applied to are normal XML documents used throughout the xmlTree API.

The xs1tStylesheet is applied to the xmlDoc with xs1tApplyStylesheet, which returns the transformed result as an xmlDoc. That document can be printed or saved like any other xmlDoc, as described in the Saving section (p. 342).

EXAMPLE
1.6. XML API

```xml
<doc = (xmlTreeNewDocFromString( 'example.xml'))>
<style = (xsltParseStylesheetString( 'example.xsl'))>
$result = (xsltApplyStylesheet($doc, $style))>
$output = (xmlTreePrintDoc($result, 'INDENT'))>
```

The result is:

```xml
$output
```

See the sample xmlTree10_XSL for examples using XSL.

SEE ALSO

xsltApplyStylesheet, xsltParseStylesheetDoc, xsltParseStylesheetFile, xsltParseStylesheetString

1.6.6 xmlTree FAQ

This section addresses some common pitfalls when working with the xmlTree API.

Text Nodes and Children

A common problem when working with the xmlTree API is getting the first child of the root element and finding has no content, despite the fact that the root’s first child element does have content. This is due to an often overlooked aspect of the XML specification:

- All whitespace that occurs between XML elements is significant.

Consider the following XML document:

```xml
<?xml version="1.0" encoding="UTF-8">
<top>
    <item>I’m an item!</item>
    <item>So am I!</item>
</top>
```

It looks like the `<top>` element as two children, both of which are `<item>` elements. But remembering that whitespace is significant, it actually has five children, which includes the text in-between the nodes:

- a text node, containing a newline and 4 spaces (the text between `<top>` and the first `<item>`) 
- an element node, the first `<item>`
- a text node, containing a newline and 4 spaces (the text between the two `<item>s`)
CHAPTER 1. TEXIS WEB SCRIPT

- an element node, the second <item>
  - a text node, containing only a newline (the text between the last <item> and the closing </top>)

Going back to the pitfall from the beginning, when we get the first child of the root element we're actually getting the text node in between the <top> and <item> elements, instead of the <item> element itself.

Without those extra text nodes, the document would look like this:

<?xml version="1.0" encoding="UTF-8">
<top><item>I’m an item!</item><item>So am I!</item></top>

With many XML documents it is handy to ignore empty whitespace and think of <top> as only having two children. This can be done by passing the option XML_PARSE_NOBLANKS when parsing the XML data. The parser will determine when a text node contains only whitespace (as defined by the XML spec), and discard them when they do.

See xmlTreeNewDocFromString() (p. 456) and xmlTreeNewDocFromFile() (p. 454) for more information.

xmlDoc vs. Root Element

The xmlDoc of an XML document is not the same as the "root element" of the XML document. The xmlDoc node is imaginary, and sits above the root element.

Consider the following XML document:

<?xml version="1.0" encoding="UTF-8">
<?xml-stylesheet type="text/xsl" href="example.xsl"?>
<!-- This comment exists outside of the root element -->
<top>
  <item>I’m an item!</item>
  <item>So am I!</item>
</top>

Here, the xmlDoc has three xmlNode children:

- The processing instruction that defines the XSL stylesheet
- A comment node
- The root element, <top>

Note that the information from the XML declaration (<?xml...>) is not stored in a child node, but within the xmlDoc itself. See xmlTreeGetVersion() (p. 444) and xmlTreeGetEncoding() (p. 423) for more information.
This shows that although it’s true that XML documents can only have one root element, an xmlDoc may have multiple children, and the root element may not be the first child of the xmlDoc.

This is why you should use xmlTreeGetRootElement() to get the root element of a document instead of xmlTreeGetFirstChild().

**The Concept of Nothing**

Vortex does not have a concept of NULL. There are many times when 'nothing' needs to be returned, such as calling

```xml
<?next = (xmlTreeGetNext($node))>
```

when there is no next for node. In these cases, it actually returns nothing (no values). This means next will be set just the same as if you had run

```xml
<?next = >
```

“Nothing” compares equal to the empty string, so it’s possible to do things like

```xml
<?node = (xmlTreeGetFirstChild($parent))>
<?while $node neq ''>
    <?name = (xmlTreeGetName($node))>
    name: $name
    <?node = (xmlTreeGetNext($node))>
</while>
```

**Returning Arrays**

Many times multiple items are returned from XML functions, such as xmlTreeGetChildren(), or xmlTreeXPathExecute(). These multiple results are put into a Vortex array, and can be looped over just like anything else.

```xml
<?children = (xmlTreeGetChildren($parent))>
<?if '' eq $children>
    No children for parent.
<?else>
    <loop $children>
        <?name = (xmlTreeGetName($children))>
        child $name is $type
    </loop>
<?if>
```

If sqlcp arrayconvert is on, which it is in version 6 and later.
Using Namespaces

Namespaces are represented in the xmlTree API as xmlNs structures, which contain prefix/URI pair for the namespace. Only one xmlNs structure exists per namespace, and all nodes that use that namespace point back to that namespace structure.

xmlTreeGetNsDef() gets the namespaces that are defined on that node, which could be more than one.
xmlTreeGetNs() gets the namespace that applies to that node, which can only be one.

Consider the following XML document:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<top xmlns:myns="urn:exampleNs"
     xmlns:otherns="urn:someOtherNs">
  <myns:item>I'm an item!</myns:item>
  <myns:item>Me too!</myns:item>
</top>
```

There are two xmlNs structure in this document, both of which are defined on the <top> element. Therefore, calling xmlTreeGetNsDef() on <top> would return one xmlNs, and the second could be retrieved by calling xmlTreeGetNext() on the first. The <top> element defines them, but does NOT use either of them (it is in the default, empty namespace), so calling xmlTreeGetNs() on <top> will return nothing.

The <item> nodes do not declare any namespaces themselves, so xmlTreeGetNsDef() will return nothing for both of them. They both use the urn:exampleNs namespace, so calling xmlTreeGetNs() will return the xmlNs for urn:exampleNs.

See xmlTree04-Ns and xmlTree05_SetNS for examples of working with namespaces in the xmlTree API.

XSL Parameter Quoting

When passing parameters values to the XSL engine that are string constants, those values must be quoted twice.

```
<$names = "lang">
<$values = "'EN'">
<$result = (xsltApplyStylesheet($doc, $style, $names, $values))>
```

The values get two sets of quotes around them, both double and single. The Vortex assignment consumes the outer double quotes, which sets the Vortex variable $values to ’EN’. When the variable is referenced in XSL using <xsl:value-of select="$lang"/>, it resolves to <xsl:value-of select="'EN'"/>. This selects ’the string constant EN’, which the desired behavior.
If the Vortex variable is set as <$values = "EN">, then Vortex consumes the quotes and the Vortex variable $values gets the value EN. When this is accessed in XSL using <xsl:value-of select="$lang"/>, it resolves to <xsl:value-of select="EN"/>. This selects 'the contents of the element <EN> in the XML document being transformed'.

### DTDs, Entities, and Entity References

The xmlTree API has limited support for reading DTDs, geared towards using entities defined in an XML document. All DTD objects are xmlNode's with various type values - XML_DTD_NODE, XML_ENTITY_DECL, etc.

It’s possible for an XML document to have two different DTDs, an internal DTD and an external DTD. The XML_DTD_NODE object from these can be fetched with the xmlTreeGetInternalSubset() and xmlTreeGetExternalSubset() functions, respectively.

Entities are declared in the DTD, and when they are used in a document, XML_ENTITY_REF nodes are used to refer back to the entities. Consider the following XML document:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE top [
<!ENTITY ts "Thunderstone Software, LLC.">]
<!ELEMENT top (#PCDATA)>]
<top>ts is &ts;!</top>
</capture>
```

This defines a single entity, ts, and it is referenced in the element top. Entities don’t have to be dealt with if you don’t want to:

- You can permanently substitute them when parsing with the XML_PARSE_NOENT option. This substitutes the entity’s value in the tree, so in the previous example <top> will only have one text child, ts is Thunderstone Software, LLC., rather than an entity reference. See xmlTreeNewDocFromFile() (p. 454) and xmlTreeNewDocFromString() (p. 456) for more information.

- calling xmlTreeGetContent() on <top> will return ts is Thunderstone Software, LLC., which performs the entity substitution for you. It can also be called with NO_INLINE to leave entity references in place. See xmlTreeGetContent() (p. 420) for more information.
The example document would have the following hierarchical structure in the xmlTree API:

```
XML_DOC_NODE
  | |
  XML_DTD_NODE
    | |
    +--XML_ENTITY_DECL "------"
    |
    \--XML_ELEMENT_DECL

\--XML_ELEMENT_NODE <top>
  | |
  +--XML_TEXT_NODE "ts is "
  |
  +--XML_ENTITY_REF
      |
      \--XML_ENTITY_DECL
  |
  \--XML_TEXT_NODE "!"
```

The element `<top>` actually has three children: the entity reference, and the two text node children around it. The entity reference appears to have a child, which just refers back to the entity that was declared in the DTD. Calling `xmlTreeGetAllContent()` on the XML_ENTITY_REF node will properly return the entity's contents.

See the sample xmlTree09-DTD for an example of working with an XML document and DTD like this.

### 1.6.7 Using the xmlReader

The xmlReader API uses a reader object that progresses through the file (advanced by `xmlReaderRead()`), stopping at points of interest (start/end elements, text contents, sub-elements, etc). The functions `xmlReaderGetType()` and `xmlReaderGetName()` are essential for figuring out where the reader object is after each `xmlReaderRead()` call.

See the sample xmlReader01_Example for examples of working with the xmlReader API.

### Opening a Reader

Unlike the xmlTree and xmlWriter APIs, a xmlReader must be started from a source, either a file or a string. Both can take parsing options to affect the data that's read.

SEE ALSO

xmlReaderNewFromFile, xmlReaderNewFromString
### Moving the Reader

For most situations there’s only one way to move the reader: forward, via `xmlReaderRead()`, which moves to the next item type.

The exception is attributes. The reader does not stop on attributes when advancing with `xmlReaderRead()`. Instead, all attributes of an element are read when the element is read. They can then be randomly accessed and iterated over as long as the reader remains on the element (which is until `xmlReaderRead()` is again called).

**SEE ALSO**

`xmlReaderRead`, `xmlReaderMoveToAttribute`, `xmlReaderMoveToAttributeNumber`, `xmlReaderMoveToFirstAttribute`, `xmlReaderMoveToNextAttribute`, `xmlReaderMoveToElement`

### Document Information

These functions provide information about the document that is specified in the XML declaration (`<?xml...`).

**SEE ALSO**

`xmlReaderGetEncoding`, `xmlReaderGetLang`, `xmlReaderGetVersion`

### Basic Information

These functions provide information about what the reader is currently on.

**SEE ALSO**

`xmlReaderGetType`, `xmlReaderGetName`, `xmlReaderGetAllContent`, `xmlReaderGetContent`

### Namespace Information

Namespace information is preserved by the `xmlReader` - when a namespace is declared at the root element, it is remembered so when a subsequent node uses that prefix, the reader can tell you what the URI for the namespace is.

**SEE ALSO**

`xmlReaderGetLocalName`, `xmlReaderGetNsURI`, `xmlReaderGetNsPrefix`
CHAPTER 1. TEXIS WEB SCRIPT

Attribute Information

These functions provide information about the attributes of an element. If you want an attribute’s value and know the name of the attribute you want, xmlReaderGetAttribute() provides a quick shortcut. Otherwise these functions, in combination with the Moving the Reader functions and Basic Information functions, provide full information about the attributes.

SEE ALSO

xmlReaderGetAttribute, xmlReaderGetAttributeCount, xmlReaderGetAttributeNumber

Other Information

These functions provide information about the state of the reader.

The parser can read ahead and buffer some of the XML message, so functions like this might not match the current node.

SEE ALSO

xmlReaderGetBytesConsumed, xmlReaderGetColumn, xmlReaderGetDepth, xmlReaderGetLine, xmlReaderIsEmptyElement

1.6.8 Using the xmlWriter

The xmlWriter operates rather simply, doing exactly what you tell it to. It’s similar to printing XML manually, except there are functions for automatically doing XML related things such as starting/ending elements, attributes, comments, etc.

See the sample xmlWriter01_Example for examples of using the xmlWriter interface.

Starting

All xmlWriter objects output to either a file (xmlWriterNewToFile()) or to an in-memory variable (xmlWriterNewToString()). You may also tell it to automatically indent the contents, with xmlWriterSetIndent().

The XML prolog (the <?xml version=...>) is created with xmlWriterStartDocument(). While not absolutely required to be well-formed XML, it is recommended that all XML documents have an XML prolog.

SEE ALSO

1.6. XML API

Finishing

Documents are finished with a `xmlWriterEndDocument()`. It closes any open tags, prevents any further accidental writes from occurring to the writer, and if the writer was created with `xmlWriterNewToFile()`, it closes the file handle.

The content of a writer created with `xmlWriterNewToString` is still available (via `xmlWriterGetContent`) after `xmlWriterEndDocument` is called.

SEE ALSO

`xmlWriterEndDocument`

Getting the Contents

If you start a writer with `xmlWriterNewToString()`, then it will write its output to an internal buffer. You can get the contents of this buffer with `xmlWriterGetContent()`. Note that unless `xmlWriterEndDocument()` has been called, the results of this may not be a complete XML document.

SEE ALSO

`xmlWriterGetContent`

Writing Output

The contents of nodes (elements, attributes, comments, etc) can be output through two functions, depending on whether the output is already escaped.

`xmlWriterWrite()` takes a string and encodes it for XML output. This properly escapes things that aren’t well-formed. Things such as Bob & Joe will be outputted as the XML-safe Bob & Joe.

`xmlWriterWriteRaw()`, on the other hand, writes exactly what you give it. This is handy if you already have a string that contains Bob & Joe, and want to write that out. `xmlWriterWrite()` would output that as Bob & Joe.

In general, use `xmlWriterWrite()`, unless you happen to have already escaped output.

SEE ALSO

`xmlWriterWrite`, `xmlWriterWriteRaw`

Below are the functions for writing various kinds of XML structures.

Elements

`xmlWriterStartElement`
`xmlWriterWrite`
`xmlWriterEndElement`
xmlWriterWriteElement

Attributes
xmlWriterStartAttribute
xmlWriterWrite
xmlWriterEndAttribute
xmlWriterWriteAttribute

Comments
xmlWriterStartComment
xmlWriterWrite
xmlWriterEndComment
xmlWriterWriteComment

Processing Instruction
xmlWriterStartPI
xmlWriterWrite
xmlWriterEndPI
xmlWriterWritePI

CDATA
xmlWriterStartCDATA
xmlWriterWrite
xmlWriterEndCDATA
xmlWriterWriteCDATA
1.6.9 xmlReaderGetAllContent

SYNOPSIS

    string xmlReaderGetAllContent(xmlReader reader)

Parameters:

    • reader - the xmlReader object

Returns:

    • text value for the current element and its sub-elements

DESCRIPTION

xmlReaderGetAllContent() gives the entire contents of this element and its sub-elements. This is in contrast to xmlReaderGetContent(), which only returns the text of the element, not its children.

xmlReaderGetAllContent can be handy for HTML-like XML data:

    <div>
        This div has <b>a lot</b> of <i>emphasis</i> in its text.
    </div>

Calling xmlReaderGetAllContent() when the reader is on <div> will give you
This div has a lot of emphasis in its text.
where as xmlReaderGetContent() would return
This div has of in its text.

EXAMPLE

    <$content = (xmlReaderGetAllContent($reader))>

CAVEATS

If you are on the opening of an element, the parser must read ahead, up to the closing of that element, in order to make sure it sees all the text nodes. In other words, if you’re working with a 2GB XML file, you probably don’t want to call xmlReaderGetAllContent() on the root node.
SEE ALSO

xmlReaderGetName, xmlReaderGetContent, xmlReaderGetAttribute
1.6. XML API

1.6.10 xmlReaderGetAttribute

SYNOPSIS

string xmlReaderGetAttribute(xmlReader reader, string name [, string nsURI])

Parameters:

- xmlReader - the xmlReader object
- name - the name of the attribute whose value you want
- nsURI (optional) - The URI or the namespace for the attribute. Defaults to any namespace.

Returns:

- The value of the attribute.

DESCRIPTION

xmlReaderGetAttribute() gets the value of a specific attribute on the current element.

If no nsURI is given, then the qualified name (ns prefix:local name) is used, and the namespace URI is not considered. If a nsURI is given, then the attribute with that local name in the nsURI namespace will be fetched, regardless of prefix.

EXAMPLE

```xml
<capture>
<?xml version="1.0"?>
<top attr1="attr1 noNS" myns:attr1="attr1 myns"
     xmlns:myns="urn:nsExample"/>
</capture>
$xml = $ret
<reader = (xmlReaderNewFromString($xml))>

<!-- move to the first element -->
$ret = (xmlReaderRead($reader))>

$val1 = (xmlReaderGetAttribute($reader, 'attr1'))>
no ns, no prefix: ($val1)
$val2 = (xmlReaderGetAttribute($reader, 'myns:attr1'))>
no ns, myns prefix: ($val2)
```
<$val3 = (xmlReaderGetAttribute($reader, 'attr1', ''))>
blank ns: ($val3)
<$val4 = (xmlReaderGetAttribute($reader, 'attr1',
    'urn:nsExample'))>
nsExample ns: ($val4)

Will produce the following output:

no ns, no prefix: (attr1 noNS)
no ns, myns prefix: (attr1 myns)
blank ns: (attr1 noNS)
nsExample ns: (attr1 myns)

SEE ALSO
xmlReaderGetAttributeCount, xmlReaderGetAttributeNumber
1.6.11 xmlReaderGetAttributeCount

SYNOPSIS

int xmlReaderGetAttributeCount(xmlReader reader)

Parameters:

- reader - the xmlReader object

Returns:

- The number of attributes on the current element

DESCRIPTION

xmlReaderGetAttributeCount() tells you the number of attributes that are on the current element.

EXAMPLE

<$attrNum = (xmlReaderGetAttributeCount($reader))>

SEE ALSO

xmlReaderGetAttribute, xmlReaderGetAttributeNumber
### 1.6.12 xmlReaderGetAttributeNumber

#### SYNOPSIS

```c
string xmlReaderGetAttributeNumber(xmlReader reader, int number)
```

**Parameters:**
- `reader` - the xmlReader object
- `number` - the index of the attribute you want (starting at 0)

**Returns:**
- The text value of the requested attribute.

#### DESCRIPTION

`xmlReaderGetAttributeNumber()` will get the text value of an attribute, based on its ordering.

It uses a 0-based numbering system - 0 is the first attribute, 1 is the second, and so on.

#### EXAMPLE

```xml
<$attrNum = (xmlReaderGetAttributeCount($reader))>
<while $loop lt $attrNum>
    <$val = (xmlReaderGetAttributeNumber($reader, $loop))>
    Attr $loop is [$val]
</while>
```

#### CAVEATS

The ordering of the attributes as provided by this function might NOT be the same as the ordering of the attributes in the XML document. This is legal, as the ordering of attributes is not significant in an XML document (see section 3.1 of the XML spec).

#### SEE ALSO

`xmlReaderGetAttribute`, `xmlReaderGetAttributeCount`
1.6. XML API

1.6.13 xmlReadergetBytesConsumed

SYNOPSIS

    int xmlReadergetBytesConsumed(xmlReader reader)

Parameters:

    • reader - the xmlReader object

Returns:

    • the number of bytes consumed

DESCRIPTION

xmlReadergetBytesConsumed() returns the number of bytes the reader has consumed so far.

EXAMPLE

    <$bytes = (xmlReadergetBytesConsumed($reader))>

CAVEATS

The xmlReader parser can read ahead and buffer some of the XML message, so functions like this might not match the current element.

SEE ALSO

xmlReaderGetColumn, xmlReaderGetDepth, xmlReaderGetLine, xmlReaderGetType, xmlReaderIsEmptyElement
1.6.14 xmlReaderGetColumn

SYNOPSIS

```
int xmlReaderGetColumn(xmlReader reader)
```

Parameters:

- **reader** - the xmlReader object

Returns:

- the current column of the parser

DESCRIPTION

xmlReaderGetBytesConsumed() returns the current column of the parser.

EXAMPLE

```
<$bytes = (xmlReaderGetColumn($reader))>
```

CAVEATS

The xmlReader parser can read ahead and buffer some of the XML message, so functions like this might not match the current element.

SEE ALSO

xmlReaderGetBytesConsumed, xmlReaderGetDepth, xmlReaderGetLine, xmlReaderGetType, xmlReaderIsEmptyElement
1.6. XML API

1.6.15 xmlReaderGetContent

SYNOPSIS

```
string xmlReaderGetContent(xmlReader reader)
```

Parameters:

- `reader` - the xmlReader object

Returns:

- The content of the current element or attribute.

DESCRIPTION

xmlReaderGetContent() returns the text content of the current node. If you are on a text node or an attribute, it will simply be their value. If you’re on an element, it will be a concatenation of all of the element’s text values.

EXAMPLE

```
<$content = (xmlReaderGetContent($reader))>
```

CAVEATS

If you are on the opening of an element, the parser must read ahead, up to the closing of that element, in order to make sure it sees all the text nodes. In other words, if you’re working with a 2GB XML file, you probably don’t want to call xmlReaderGetContent() on the root node.

SEE ALSO

xmlReaderGetName, xmlReaderGetAllContent
1.6.16 xmlReaderGetDepth

SYNOPSIS

    int xmlReaderGetDepth(xmlReader reader)

Parameters:

- reader - the xmlReader object

Returns:

- The reader’s current depth

DESCRIPTION

xmlReaderGetDepth() returns how many elements deep the reader’s current position is, where the root element is 1, its children are 2, its children’s children are 3, etc.

EXAMPLE

    <$depth = (xmlReaderGetDepth($reader))>

CAVEATS

The xmlReader parser can read ahead and buffer some of the XML message, so functions like this might not match the current element.

SEE ALSO

xmlReaderGetBytesConsumed, xmlReaderGetColumn, xmlReaderGetLine, xmlReaderGetType, xmlReaderIsEmptyElement
1.6.17  xmlReaderGetEncoding

SYNOPSIS

    string xmlReaderGetEncoding(xmlReader reader)

Parameters:

    • reader - the xmlReader object

Returns:

    • The encoding of the document.

DESCRIPTION

xmlReaderGetEncoding() returns the character encoding used in the document, as set by the XML declaration, such as:

    <?xml version="1.0" encoding="ISO-8859-1"?>

If no encoding is given in a declaration (or no declaration is present), then xmlReaderGetEncoding() will return nothing (although UTF-8 is assumed as the encoding, as defined in the XML spec).

EXAMPLE

    <$encoding = (xmlReaderGetEncoding($reader))>

SEE ALSO

xmlReaderGetLang, xmlReaderGetVersion
1.6.18  xmlReaderGetLang

SYNOPSIS

```c
string xmlReaderGetLang(xmlReader reader)
```

Parameters:

- `reader` - the xmlReader object

Returns:

- the value of the special `xml:lang` attribute for the current element (if any).

DESCRIPTION

`xml:lang` is a special attribute defined by the XML spec for defining the language of elements.

`xmlReaderGetLang()` returns the defined language for the current element, whether it's defined there or inherited from a parent.

See the Language Identification section of the XML spec for more information:

http://www.w3.org/TR/REC-xml/#sec-lang-tag

EXAMPLE

```xml
<$lang = (xmlReaderGetLang($reader))>
```

SEE ALSO

`xmlReaderGetEncoding`, `xmlReaderGetVersion`
1.6. XML API

1.6.19 xmlReaderGetLine

SYNOPSIS

    int xmlReaderGetLine(xmlReader reader)

Parameters:

  • reader - the xmlReader object

Returns:

  • the current line of the parser

DESCRIPTION

xmlReaderGetLine() returns the current line of the XML file that the parser is on.

EXAMPLE

    <$line = (xmlReaderGetLine($reader))>

CAVEATS

The xmlReader parser can read ahead and buffer some of the XML message, so functions like this might not match the current element.

SEE ALSO

xmlReaderGetBytesConsumed, xmlReaderGetColumn, xmlReaderGetDepth, xmlReaderGetType, xmlReaderIsEmptyElement
1.6.20 xmlReaderGetLocalName

SYNOPSIS

    string xmlReaderGetLocalName(xmlReader reader)

Parameters:

    • reader - the xmlReader object

Returns:

    • the local name of the current element or attribute

DESCRIPTION

xmlReaderGetLocalName() returns the local name of the current element or attribute, which is the name without any namespace prefix on it.

For example, calling xmlReaderGetLocalName() on the elements <item> and <myns:item> will both return item. It is useful for comparing things in a namespace-aware manner.

EXAMPLE

    <$localname = (xmlReaderGetLocalName($reader))>

SEE ALSO

xmlReaderGetNsURI, xmlReaderGetNsPrefix
1.6.21 xmlReaderGetName

SYNOPSIS

    string xmlReaderGetName(xmlReader reader)

Parameters:

- reader - the xmlReader object

Returns:

- the full name of the current element or attribute

DESCRIPTION

xmlReaderGetName() returns the full name of the element or attribute the reader is currently on. This includes any namespace information, so calling it on <item> returns item, and calling it on <myns:item> returns myns:item. The prefix and local name can be retrieved individually through xmlReaderGetNsPrefix() and xmlReaderGetLocalName(), respectively.

EXAMPLE

    <$name = (xmlReaderGetName($reader))>

SEE ALSO

xmlReaderGetAllContent, xmlReaderGetContent
1.6.22 xmlReaderGetNsURI

SYNOPSIS

    string xmlReaderGetNsURI(xmlReader reader)

Parameters:

- reader - the xmlReader object

Returns:

- the URI for the namespace of the current element or node

DESCRIPTION

xmlReaderGetNsURI() determines and returns the URI for the namespace of the current element or attribute. Inherited namespaces work properly - Namespaces that are defined in the root element of a document will be resolved if xmlReaderGetNsURI() is called on a sub-element that uses that namespace.

EXAMPLE

    <$uri = (xmlReaderGetNsURI($reader))>

SEE ALSO

xmlReaderGetLocalName, xmlReaderGetNsPrefix
1.6.23 xmlReaderGetNsPrefix

SYNOPSIS

    string xmlReaderGetNsPrefix(xmlReader reader)

Parameters:

    • reader - the xmlReader object

Returns:

    • the namespace prefix of the current element or attribute

DESCRIPTION

xmlReaderGetNsPrefix() returns the namespace prefix on the current element or attribute (if any). Calling it on `<item>` will return an empty string, but calling it on `<myns:item>` will return `myns`.

EXAMPLE

    <$prefix = (xmlReaderGetNsPrefix($reader))>

SEE ALSO

xmlReaderGetLocalName, xmlReaderGetNsURI
1.6.24 xmlReaderGetType

SYNOPSIS

    string xmlReaderGetType(xmlReader reader)

Parameters:

- reader - the xmlReader object

Returns:

- the type of the current node

DESCRIPTION

xmlReaderGetType() returns the type of whatever it is that the xmlReader is currently on. The possible values are:

- XML_ELEMENT_NODE
- XML_ATTRIBUTE_NODE
- XML_TEXT_NODE
- XML_CDATA_NODE
- XML_ENTITY_REF_NODE
- XML_ENTITY_NODE
- XML_PI_NODE
- XML_COMMENT_NODE
- XML_DOCUMENT_NODE
- XML_DOCUMENT_TYPE_NODE
- XML_DOCUMENT_FRAG_NODE
- XML_DECLARATION_NODE
- XML_WHITESPACE_NODE
- XML_SIGNIFICANT_WHITESPACE_NODE
- XML_END_ELEMENT_NODE
- XML_END_ENTITY_NODE
- XML_DECLARATION_NODE
EXAMPLE

<$type = (xmlReaderGetType($reader))>

SEE ALSO

xmlReaderGetBytesConsumed, xmlReaderGetColumn, xmlReaderGetDepth,
xmlReaderGetLine, xmlReaderIsEmptyElement
1.6.25  xmlReaderGetVersion

SYNOPSIS

    string xmlReaderGetVersion(xmlReader reader)

Parameters:

  • reader - the xmlReader object

Returns:

  • the XML version of the data

DESCRIPTION

xmlReaderGetVersion() returns the XML version used by the XML data, as defined in the XML declaration:

    <?xml version="1.0" encoding="UTF-8"?>
    <top>
          ...
    </top>

For this data, xmlReaderGetVersion() would return 1.0.

EXAMPLE

    <$version = (xmlReaderGetVersion($reader))>

SEE ALSO

xmlReaderGetEncoding, xmlReaderGetLang
1.6.26 xmlReaderIsEmptyElement

SYNOPSIS

    int xmlReaderIsEmptyElement(xmlReader reader)

Parameters:

- reader - the xmlReader object

Returns:

- Whether the current element is an empty (<p/>) element

DESCRIPTION

xmlReaderIsEmptyElement() returns 1 if the current element is empty, 0 if not and -1 in case of error. These are sometimes called “standalone” or “self closing” elements.

More precisely, this is for testing whether elements will have an end element. E.g. <a/> will be considered empty while <a></a> will not.

EXAMPLE

    <$isEmpty = (xmlReaderIsEmptyElement($reader))>

CAVEATS

Invoke xmlReaderIsEmptyElement when you’re on the XML_READER_TYPE_ELEMENT to tell whether it’s an opening element of a balanced pair or an empty element. If the reader is on XML_READER_TYPE_END_ELEMENT, then xmlReaderIsEmptyElement will always return 0, as you’re on the closing element of a non-empty pair.

SEE ALSO

xmlReaderGetColumn, xmlReaderGetDepth, xmlReaderGetLine, xmlReaderGetType
1.6.27 xmlReaderMoveToAttribute

SYNOPSIS

    int xmlReaderMoveToAttribute(xmlReader reader, string name)

Parameters:

• reader - the xmlReader object
• name - the name of the attribute you want to move to

Returns:

• 1 - success
• 0 - attribute not found
• -1 - other error

DESCRIPTION

xmlReaderMoveToAttribute() moves the parser on to the specified attribute of this function. This makes functions that operate on the "current node" (such as xmlReaderGetName() and xmlReaderGetContent()) take effect on the attribute, rather than the element itself. If the attribute is not found, 0 is returned and the readers stays on the element.

EXAMPLE

    <$ret = (xmlReaderMoveToAttribute($reader, 'myAttr'))>

SEE ALSO

xmlReaderMoveToAttributeNumber, xmlReaderMoveToElement, xmlReaderMoveToFirstAttribute, xmlReaderMoveToNextAttribute, xmlReaderRead
1.6.28 xmlReaderMoveToAttributeNumber

SYNOPSIS

    int xmlReaderMoveToAttributeNumber(xmlReader reader,
                                    int number)

Parameters:

    • reader - the xmlReader object
    • number - the index of the attribute you want to move to (starting at 0)

Returns:

    • 1 - success
    • 0 - attribute not found
    • -1 - other error

DESCRIPTION

xmlReaderMoveToAttributeNumber() moves the reader to an attribute of the current element, based on its ordering.
It uses a 0-based numbering system - 0 is the first attribute, 1 is the second, and so on.

EXAMPLE

    <$attrNum = (xmlReaderGetAttributeCount($reader))>
    <while $loop lt $attrNum>
        <$val = (xmlReaderGetAttributeNumber($reader, $loop))>
        Attr $loop is [$val]
    </while>

CAVEATS

The ordering of the attributes as provided by this function might NOT be the same as the ordering of the attributes in the XML document. This is legal, as the ordering of attributes is not significant in a XML document (see section 3.1 of the XML spec).
SEE ALSO

xmlReaderMoveToAttribute, xmlReaderMoveToElement,
xmlReaderMoveToFirstAttribute, xmlReaderMoveToNextAttribute, xmlReaderRead
1.6.29  xmlReaderMoveToElement

SYNOPSIS

    int xmlReaderMoveToElement(xmlReader reader)

Parameters:

    • reader - the xmlReader object

>Returns:

    • 1 - success
    • 0 - not moved
    • −1 - other error

DESCRIPTION

xmlReaderMoveToElement() moves the reader back on to the element after it has been moved on to
one of its attributes.

If you want to advance the reader to the next node in the XML document, see xmlReaderRead().

EXAMPLE

    <$ret = (xmlReaderMoveToElement($reader))>

CAVEATS

The ordering of the attributes as provided by this function might NOT be the same as the ordering of the
attributes in the XML document. This is legal, as the ordering of attributes is not significant in an XML
document (see section 3.1 of the XML spec).

SEE ALSO

xmlReaderMoveToAttribute, xmlReaderMoveToAttributeNumber, xmlReaderMoveToFirstAttribute, xmlReaderMoveToNextAttribute, xmlReaderRead
1.6.30 xmlReaderMoveToFirstAttribute

SYNOPSIS

```
int xmlReaderMoveToFirstAttribute(xmlReader reader)
```

Parameters:

- `reader` - the `xmlReader` object

Returns:

- 1 - success
- 0 - not found
- -1 - other error

DESCRIPTION

`xmlReaderMoveToFirstAttribute()` moves the `xmlReader` on to the first attribute of the current element, which makes it the target of functions like `xmlReaderGetName()` and `xmlReaderGetContent()`.

EXAMPLE

```xml
<$ret = (xmlReaderMoveToFirstAttribute($reader))>
<while $ret eq 1>
    <$name = (xmlReaderGetName($reader))>
    <$content = (xmlReaderGetContent($reader))>
    attr '$name' = '$content'
    <$ret = (xmlReaderMoveToNextAttribute($reader))>
</while>
```

CAVEATS

The ordering of the attributes as provided by this function might NOT be the same as the ordering of the attributes in the XML document. This is legal, as the ordering of attributes is not significant in an XML document (see section 3.1 of the XML spec).
SEE ALSO

xmlReaderMoveToAttribute, xmlReaderMoveToAttributeName,
xmlReaderMoveToElement, xmlReaderMoveToFirstChild,
xmlReaderMoveToNextAttribute, xmlReaderRead
### 1.6.31 xmlReaderMoveToNextAttribute

**SYNOPSIS**

```c
int xmlReaderMoveToNextAttribute(xmlReader reader)
```

**Parameters:**
- `reader` - the xmlReader object

**Returns:**
- `1` - success
- `0` - not found
- `−1` - other error

**DESCRIPTION**

`xmlReaderMoveToNextAttribute()` advances the xmlReader to the next attribute of the element.

If the reader is currently on an element, it is moved to the first attribute.

**EXAMPLE**

```xml
<$ret = (xmlReaderMoveToFirstAttribute($reader))>
<while $ret eq 1>
    <$name = (xmlReaderGetName($reader))>
    <$content = (xmlReaderGetContent($reader))>
    attr '$name' = '$content'
    <$ret = (xmlReaderMoveToNextAttribute($reader))>
</while>
```

**CAVEATS**

The ordering of the attributes as provided by this function might NOT be the same as the ordering of the attributes in the XML document. This is legal, as the ordering of attributes is not significant in an XML document (see section 3.1 of the XML spec).

**SEE ALSO**

`xmlReaderMoveToAttribute`, `xmlReaderMoveToAttributeValue`, `xmlReaderMoveToElement`, `xmlReaderMoveToFirstAttribute`, `xmlReaderRead`
1.6.32  xmlReaderNewFromFile

SYNOPSIS

    xmlReader xmlReaderNewFromFile(string filename
         [, string encoding ]
         [, string options ] )

Parameters:

- filename - the name of the XML file you’d like to read
- encoding (optional) - encoding to use for the file. If none given, xmlReader will attempt to
discover on its own.
- options (optional) - a comma-separated list of parsing options (see below)

Returns:

- the xmlReader for this file

DESCRIPTION

xmlReaderNewFromFile() opens a new xmlReader on the filename specified. The entire XML file
is not read when the xmlReader is opened, allowing for larger files to be read without loading them
entirely in memory.

The possible values for the options parameter are:

- XML_PARSE_RECOVER - recover on errors
- XML_PARSE_NOENT - substitute entities
- XML_PARSE_DTDLOAD - load the external subset
- XML_PARSE_DTDATTR - default DTD attributes
- XML_PARSE_DTDVALID - validate with the DTD
- XML_PARSE_NOERROR - suppress error reports
- XML_PARSE_NOWARNING - suppress warning reports
- XML_PARSE_PEDANTIC - pedantic error reporting
- XMLPARSE_NOBLANKS - remove blank nodes
- XMLPARSE_XINCLUDE - Implement XInclude substitution
• XML_PARSE_NONET - Forbid network access
• XML_PARSE_NSNCLEAN - remove redundant namespaces declarations
• XML_PARSE_NOCDATA - merge CDATA as text nodes
• XML_PARSE_NOXINCNODE - do not generate XINCLUDE START/END nodes

EXAMPLE

<!-- read the file, but ignore blank text nodes
== and load the DTD (as long as it doesn’t
== involve a network fetch). -->
<$reader = (xmlReaderNewFromFile($filename,
'XML_PARSE_NOBLANKS, XML_PARSE_DTDLOAD, XML_PARSE_NONET'))>

SEE ALSO
xmlReaderNewFromString
1.6.33  xmlReaderNewFromString

SYNOPSIS

    xmlReader xmlReaderNewFromString(string data
           [, string encoding ]
           [, string options ] )

Parameters:

- data - the XML you want to parse and read
- encoding (optional) - encoding to use for the file
- options (optional) - a comma-separated list of parsing options (see below)

Returns:

- the xmlReader for this string

DESCRIPTION

xmlReaderNewFromString() parses XML data from a Vortex variable into an xmlReader. A copy of the variable is made at the time that xmlReaderNewFromString() is called, so changes to the variable that contains XML data after the xmlReader has been created will not affect the xmlReader.

The possible values for the options parameter are:

- XML_PARSE_RECOVER - recover on errors
- XML_PARSE_NOENT - substitute entities
- XML_PARSE_DTDLOAD - load the external subset
- XML_PARSE_DTDATTR - default DTD attributes
- XML_PARSEDTDVALID - validate with the DTD
- XML_PARSE_NOERROR - suppress error reports
- XML_PARSE_NOWARNING - suppress warning reports
- XML_PARSE_PEDANTIC - pedantic error reporting
- XML_PARSE_NOBLANKS - remove blank nodes
• XML_PARSE_XINCLUDE - Implement XInclude substitution
• XML_PARSE_NONET - Forbid network access
• XML_PARSE NSCLEAN - remove redundant namespaces declarations
• XML_PARSE_NOCDATA - merge CDATA as text nodes
• XML_PARSE_NOXINCNODE - do not generate XINCLUDE START/END nodes

EXAMPLE

```php
$xml = "<item>I’m a simple XML document!</item>"
$xmlReader = (xmlReaderNewFromString($xml))
```

SEE ALSO

xmlReaderNewFromFile
1.6.34 xmlReaderRead

SYNOPSIS

    int xmlReaderRead(xmlReader reader)

Parameters:

    • reader - the xmlReader object

Returns:

    • 1 - the next node was read successfully
    • 0 - there are no more nodes to read
    • -1 - error

DESCRIPTION

xmlReaderRead() advances the xmlReader to the next node in the XML data.

EXAMPLE

    <while (xmlReaderRead($reader) = '1')>
        <$name = (xmlReaderGetName($reader))>
        <$type = (xmlReaderGetType($reader))>
        On $name ($type)...
    </while>

SEE ALSO

xmlReaderMoveToAttribute, xmlReaderMoveToAttributeNumber, xmlReaderMoveToElement, xmlReaderMoveToFirstAttribute, xmlReaderMoveToNextAttribute
1.6.35 xmlTreeAddChild

SYNOPSIS

 xmlNode xmlTreeAddChild(xmlNode parent, xmlNode child)

Parameters:

• parent - The node that you want to assign the child to.

• child - The node that you want to move to being a child of parent.

Returns:

• the child

DESCRIPTION

xmlTreeAddChild() adds the node child as a child of parent. If it was already in an XML
document, it is fully removed from its original location. The child is appended at the end of the list of the
parent’s children.

EXAMPLE

Given the XML document

<top>
   <item>item 1</item>
   <item>item 2</item>
   <summary>
      <name>Squiggy</name>
      <detail>fuzzy</detail>
      <color>blue</color>
      <size>big</size>
   </summary>
</top>

The command

<$ret = (xmlTreeAddChild($top, $detail))>
Would result in:

```xml
<top>
  <item item 1</item>
  <item item 2</item>
  <summary>
    <name>Squiggy</name>
    <color>blue</color>
    <size>big</size>
  </summary>
  <detail>fuzzy</detail>
</top>
```

CAVEATS

If this operation results in two text nodes being siblings of each other, the second node will be merged into the first and then unlinked. Any changes made to the second text node will not be reflected in the tree.

SEE ALSO

xmlTreeAddChildList, xmlTreeAddNextSibling, xmlTreeAddPrevSibling, xmlTreeAddSibling, xmlTreeSetRootElement
1.6.36 xmlTreeAddChildList

SYNOPSIS

xmlNode xmlTreeAddChildList(xmlNode parent, xmlNode child)

Parameters:

• parent - The node that you want to assign the child to

• child - The first node in the list of children that you want to assign to parent

Returns:

• the child

DESCRIPTION

xmlTreeAddChildList() assigns the child and the rest of its siblings as children of the parent.

EXAMPLE

Given the XML document

```
<top>
    <item>item 1</item>
    <item>item 2</item>
    <summary>
        <name>Squiggy</name>
        <detail>fuzzy</detail>
        <color>blue</color>
        <size>big</size>
    </summary>
</top>
```

The command

```
<$ret = (xmlTreeAddChildList($top, $detail))>
```
1.6. XML API

Would result in:

```xml
<top>
  <item>item 1</item>
  <item>item 2</item>
  <summary>
    <name>Squiggy</name>
  </summary>
  <detail>fuzzy</detail>
  <color>blue</color>
  <size>big</size>
</top>
```

CAVEATS

If this operations results in two text nodes being siblings of each other, the second node will be merged into the first and then unlinked. Any changes made to the second text node will not be reflected in the tree.

SEE ALSO

xmlTreeAddChild, xmlTreeAddChildList, xmlTreeAddNextSibling, xmlTreeAddPrevSibling, xmlTreeAddSibling, xmlTreeSetRootElement
1.6.37 xmlTreeAddContent

SYNOPSIS

xmlNode xmlTreeAddContent(xmlNode node, string content)

Parameters:

• node - the xmlNode you want to add content to
• content - the text content you want to add.

Returns:

• the node

DESCRIPTION

xmlTreeAddContent() adds text to an xmlNode. If the xmlNode is an element and its last child is not a text node, one is created. Otherwise the content is appended to the last text node.

EXAMPLE

<$ret = (xmlTreeAddContent($node, ’...all the way home.’))>

SEE ALSO

xmlTreeSetName, xmlTreeSetContent
1.6.38 xmlTreeAddNextSibling

SYNOPSIS

xmlNode xmlTreeAddNextSibling(xmlNode node, xmlNode sibling)

Parameters:

- node - the xmlNode that the sibling is being added next to
- sibling - the xmlNode that is being added as the next sibling of node

Returns:

- the sibling

DESCRIPTION

xmlTreeAddNextSibling() adds sibling after the specified node.

EXAMPLE

Given the XML document

```xml
<top>
  <item>item 1</item>
  <item>item 2</item>
  <summary>
    <name>Squiggy</name>
    <detail>fuzzy</detail>
    <color>blue</color>
    <size>big</size>
  </summary>
</top>
```

The command

```xml
<$ret = (xmlTreeAddNextSibling($summary, $detail))>
```
Would result in:

```xml
<top>
  <item>item 1</item>
  <item>item 2</item>
  <summary>
    <name>Squiggy</name>
    <color>blue</color>
    <size>big</size>
  </summary>
  <detail>fuzzy</detail>
</top>
```

CAVEATS

If this operations results in two text nodes being siblings of each other, the second node will be merged into the first and then unlinked. Any changes made to the second text node will not be reflected in the tree.

SEE ALSO

xmlTreeAddChild, xmlTreeAddChildList, xmlTreeAddPrevSibling, xmlTreeAddSibling, xmlTreeSetRootElement
1.6.39 xmlTreeAddPrevSibling

SYNOPSIS

```
xmNode xmlTreeAddPrevSibling(xmlNode node, xmlNode sibling)
```

Parameters:

- `node` - the xmlNode that the sibling is being added next to
- `sibling` - the xmlNode that is being added as the previous sibling of `node`

Returns:

- the sibling

DESCRIPTION

xmlTreeAddNextSibling() adds sibling before the specified node.

EXAMPLE

Given the XML document

```
<top>
    <item>item 1</item>
    <item>item 2</item>
    <summary>
        <name>Squiggy</name>
        <detail>fuzzy</detail>
        <color>blue</color>
        <size>big</size>
    </summary>
</top>
```

The command

```
<$ret = (xmlTreeAddPrevSibling($summary, $detail))>
```
Would result in:

```xml
<top>
  <item>item 1</item>
  <item>item 2</item>
  <detail>fuzzy</detail>
  <summary>
    <name>Squiggy</name>
    <color>blue</color>
    <size>big</size>
  </summary>
</top>
```

CAVEATS

If this operations results in two text nodes being siblings of each other, the second node will be merged into the first and then unlinked. Any changes made to the second text node will not be reflected in the tree.

SEE ALSO

xmlTreeAddChild, xmlTreeAddChildList, xmlTreeAddNextSibling, xmlTreeAddSibling, xmlTreeSetRootElement
1.6.40 xmlTreeAddSibling

SYNOPSIS

```xml
xmlNode xmlTreeAddSibling(xmlNode node, xmlNode sibling)
```

Parameters:

- **node** - the xmlNode that the sibling is being added next to
- **sibling** - the xmlNode that is being added to the node’s list of siblings.

Returns:

- the sibling

DESCRIPTION

xmlTreeAddSibling() adds sibling as a sibling of node. This is the same as calling xmlTreeAddChild() using node's parent as the parent parameter.

EXAMPLE

Given the XML document

```xml
<top>
  <item>item 1</item>
  <item>item 2</item>
  <summary>
    <name>Squiggy</name>
    <detail>fuzzy</detail>
    <color>blue</color>
    <size>big</size>
  </summary>
  <other>stuff</other>
</top>
```

The command

```xml
<$ret = (xmlTreeAddSibling($summary, $detail))>
```
Would result in:

```xml
<top>
  <item>item 1</item>
  <item>item 2</item>
  <summary>
    <name>Squiggy</name>
    <color>blue</color>
    <size>big</size>
  </summary>
  <detail>fuzzy</detail>
  <other>stuff</other>
</top>
```

**CAVEATS**

If this operation results in two text nodes being siblings of each other, the second node will be merged into the first and then unlinked. Any changes made to the second text node will not be reflected in the tree.

**SEE ALSO**

`xmlTreeAddChild`, `xmlTreeAddChildList`, `xmlTreeAddNextSibling`, `xmlTreeAddSibling`, `xmlTreeSetRootElement`
1.6. XML API

1.6.41 xmlTreeCleanup

SYNOPSIS

void xmlTreeCleanup(xmlDoc doc [, int options])

Parameters:

- doc - the xmlDoc to remove unlinked orphans from
- options - specifying 1 will cause verbose messages to be printed about what it’s cleaning up

DESCRIPTION

xmlTreeCleanup() can be used in very specific circumstances to assist with memory management of XML trees. Most code will never need to call this.

If you have a situation where:

- You have a long running Vortex process, that
- interacts with a single xmlDoc over a long period of time and
- repeatedly creates and unlinks nodes via xmlTreeUnlinkNode

then the memory used by Vortex can steadily climb, as unlinked nodes are normally not freed until the xmlDoc they came from is freed. Calling xmlTreeCleanup() forces the xmlDoc to free any unlinked nodes that do not have any Vortex variables set to them.

EXAMPLE

<$ret = (xmlTreeCleanup($doc))>
1.6.42 xmlTreeClearNs

SYNOPSIS

```c
int xmlTreeClearNs(xmlNs ns)
```

Parameters:

- `ns` - the namespace to clear

Returns:

- 0 - success
- <0 - error

DESCRIPTION

`xmlTreeClearNs()` clears the prefix and URI of a namespace. It is not technically deleted, but a cleared namespace does not show up at all in the output. It can be made to show again by setting the prefix or URI.

EXAMPLE

Given the XML document

```xml
<foo:rootNode xmlns:foo="http://www.example.com/foo"
               xmlns:bar="http://www.example.com/bar">
  <foo:item>a foo-y item</foo:item>
  <bar:item>a bar-y item with different flavor</bar:item>
</foo:rootNode>
```

and `$bar` is the `<bar>` element, then calling

```c
<$ns = (xmlTreeGetNs($bar))>
<$ret = (xmlTreeClearNs($ns))>
```

and printing the document will give:

```xml
<foo:rootNode xmlns:foo="http://www.example.com/foo"
               xmlns:bar="http://www.example.com/bar">
  <foo:item>a foo-y item</foo:item>
  <item>a bar-y item with different flavor</item>
</foo:rootNode>
```
CAVEATS

Clearing a namespace and then setting a prefix without setting a URI will result in an XML document that is not well-formed. For example, starting from the following XML document:

```xml
<myns:top xmlns:myns="urn:MyNamespace"/>
```

If you can `xmlTreeClearNs()` and set the prefix to `new` (via `xmlTreeSetPrefix()`), you will end up with:

```xml
<new:top/>
```

This is not a well-formed XML document because the `new` namespace is being used without being declared.

SEE ALSO

`xmlTreeSetNsPrefix`, `xmlTreeSetNsURI`
1.6.43 xmlTreeCopyAttribute

SYNOPSIS

```c
xmlNode xmlTreeCopyAttribute(xmlNode target,
                            xmlNode attribute)
```

Parameters:

- `target` - the element that will receive the attribute
- `attribute` - the attribute to be copied

Returns:

- the newly created attribute on `target`

DESCRIPTION

`xmlTreeCopyAttribute()` copies an attribute from its original location on to another node. It leaves the original attribute intact, creating a duplicate of it for placing on `target`.

EXAMPLE

```xml
<$newAttr = (xmlTreeCopyAttribute($target, $attr))>
```

SEE ALSO

`xmlTreeCopyDoc`, `xmlTreeCopyNode`, `xmlTreeCopyNodeList`, `xmlTreeCopyAttribute`, `xmlTreeCopyAttributeList`
1.6.44  xmlTreeCopyAttributeList

SYNOPSIS

```
xmlNode xmlTreeCopyAttributeList(xmlNode target,
                               xmlNode attribute)
```

Parameters:

- **target** - the xmlNode that will receive the attribute list
- **attribute** - the first attribute in the list to be copied

Returns:

- the first attribute in the newly created list on target

DESCRIPTION

xmlTreeCopyAttributeList() copies an attribute and all following attributes from their original to a new location. Note that the attributes may appear in a different order than they do in the XML source.

xmlTreeCopyAttributeList() is often used to copy all attributes from an element by using it in conjunction with xmlTreeGetFirstAttribute().

EXAMPLE

```
<$newAttr = (xmlTreeCopyAttributeList($target, $attr))>
```

CAVEATS

The ordering of the attributes as provided by this function might NOT be the same as the ordering of the attributes in the XML document. This is legal, as the ordering of attributes is not significant in an XML document (see section 3.1 of the XML spec).

SEE ALSO

xmlTreeCopyDoc, xmlTreeCopyNode, xmlTreeCopyNodeList, xmlTreeCopyAttribute
1.6.45 xmlTreeCopyDoc

SYNOPSIS

xmlDoc xmlTreeCopyDoc(xmlDoc doc)

Parameters:

• doc - the xmlDoc to copy

Returns:

• the new xmlDoc

DESCRIPTION

xmlTreeCopyDoc() makes a full copy of a parsed XML document.

EXAMPLE

<$newDoc = (xmlTreeCopyDoc($doc))>

SEE ALSO

xmlTreeCopyNode, xmlTreeCopyNodeList, xmlTreeCopyAttribute,
xmlTreeCopyAttributeList
1.6.46 xmlTreeCopyNode

SYNOPSIS

xmlNode xmlTreeCopyNode(xmlDoc targetDoc, xmlNode node
[, int recursive])

Parameters:

- targetDoc - the xmlDoc that the new node/nodes will be placed into. This is only to help optimize 
  internal string operations. It’s possible to specify no doc by passing in an empty string (’’). If node 
  is going in the same document, it’s common to use xmlTreeGetDoc(node) as this parameter.

- node - the xmlNode to be copied

- recursive (optional) - If the node to be copied is an element (as opposed to a comment, CDATA, 
  etc.), setting recursive to 1 will cause all child nodes of the element to be copied too.
  recursive is 0 by default.

Returns:

- the new copy of node

DESCRIPTION

xmlTreeCopyNode() makes a copy of node (and its children if recursive is asserted). The 
newly-created copy is unlinked and needs to be inserted into a tree, using things like 
xmlTreeAddChild() or xmlTreeAddSibling().

If you know where the copy is going, you can specify the target document with the targetDoc parameter 
to help optimize internal string operations. Nothing will be broken if a different doc is specified in 
targetDoc than the one it ends up in once it is inserted.

EXAMPLE

<$newNode = (xmlTreeCopyNode(xmlTreeGetDoc($node), $node))>

SEE ALSO

xmlTreeCopyDoc, xmlTreeCopyNodeList, xmlTreeCopyAttribute,
xmlTreeCopyAttributeList
xmlTreeCopyNodeList

SYNOPSIS

xmlNode xmlTreeCopyNodeList(xmlDoc targetDoc, xmlNode node)

Parameters:

- targetDoc - the xmlDoc that the new node/nodes will be placed into. This is only to help optimize internal string operations. It’s possible to specify no doc by passing in an empty string (''). If node is going in the same document, it’s common to use xmlTreeGetDoc(node) as this parameter.

- node - the first xmlNode of the node list to be copied

Returns:

- the first xmlNode of the new node list

DESCRIPTION

xmlTreeCopyNodeList() creates a copy of a given node, its children (if they exist), and any of its further siblings.

The newly-created copy is unlinked and needs to be inserted into a tree, using things like xmlTreeAddChild() or xmlTreeAddSibling().

If you know where the copy is going, you can specify the target document with the targetDoc parameter to help optimize internal string operations. Nothing will be broken if a different doc is specified in targetDoc than the one it ends up in once it is inserted.

EXAMPLE

<$newNode = (xmlTreeCopyNodeList(xmlTreeGetDoc($node), $node))>

SEE ALSO

xmlTreeCopyDoc, xmlTreeCopyNode, xmlTreeCopyAttribute, xmlTreeCopyAttributeList
1.6.48 xmlTreeDumpNode

SYNOPSIS

string xmlTreeDumpNode(xmlNode node [, string options])

Parameters:

• node - the node to dump
• options - content options (see below)

Returns:

• the XML contents of node

DESCRIPTION

xmlTreeDumpNode() gives the raw XML content of node. This includes the element itself, its attributes, and all of its element and text children.

This differs from other functions like xmlTreeGetContent in that it includes the XML markup, which may be desirable depending on your needs. See the example for comparisons.

options can be a comma-separated list of:

• INDENT - add extra whitespace text nodes to indent the output.
• INNER - if specified, will not include the element itself, only children text and elements (see example below).

EXAMPLE

<capture>
<data>
  <quote>some <b class="query">test</b> content</quote>
</data>
</capture><$xml=$ret>
<$doc = (xmlTreeNewDocFromString($xml))>
<$root = (xmlTreeGetRootElement($doc))>
<$quote = (xmlTreeGetChildren($root, 'quote'))>
<$ret = (xmlTreeGetContent($quote))>
Produce

xmlTreeGetContent:
some content

xmlTreeGetAllContent:
some test content

xmlTreeDumpNode:
<quote>some <b class="query">test</b> content</quote>

xmlTreeDumpNode inner:
some <b class="query">test</b> content

SEE ALSO

xmlTreeGetName, xmlTreeGetAllContent, xmlTreeGetChildrenContent,
xmlTreeGetLine, xmlTreeGetType, xmlTreeIsBlankNode,
xmlTreeGetAttributeContent
1.6. XML API

1.6.49 xmlTreeGetAllContent

SYNOPSIS

```c
string xmlTreeGetAllContent(xmlNode node)
```

Parameters:

- node - the xmlNode to get content from

Returns:

- The text value of node and all children

DESCRIPTION

xmlTreeGetAllContent() gives the entire contents of this node and its children nodes (if they exist). This is in contrast to xmlTreeGetContent(), which only returns the text of the element, not its children.

xmlTreeGetAllContent() can be handy for things like HTML data:

```html
<div>This has <i>emphasis</i> in its text.</div>
```

Calling xmlTreeGetAllContent() on the `<div>` will give you

This has emphasis in its text.

where as xmlTreeGetContent() would return

This has in its text.

EXAMPLE

```php
<$content = (xmlTreeGetAllContent($node))>
```

SEE ALSO

xmlTreeDumpNode, xmlTreeGetNode, xmlTreeGetContent, xmlTreeGetLine, xmlTreeGetType, xmlTreeIsBlankNode, xmlTreeGetAttributeContent
xmlTreeGetAttributeContent

SYNOPSIS

```
string xmlTreeGetAttributeContent(xmlNode element, string name
[, string ns_URI])
```

Parameters:

- `element` - the element that contains the attribute you want the value of
- `name` - the name of the attribute that you want the value of
- `ns_URI` *(optional)* - the namespace URI of the attribute you’d like to retrieve (defaults to any namespace)

Returns:

- the value of the attribute `name` on the element

DESCRIPTION

`xmlTreeGetAttributeContent()` gets the content of an attribute on an element, based on the name of the attribute. It is essentially a shortcut for the combination of `xmlTreeGetAttributes()` to get the attribute `name`, and then `xmlTreeGetContent()` to get its content.

If no `ns_URI` is specified, then the namespace prefix/URIs are disregarded and only local names are compared. If a `ns_URI` is provided, only an attribute that matches a name and the namespace URI will be returned. This includes the empty namespace; if you pass in a `ns_URI` as `' '`, then only the attribute matching that name and without a namespace will be returned.

EXAMPLE

```
<$content = (xmlTreeGetAttributeContent($node, 'myattr'))>
```

CAVEATS

`xmlTreeGetAttributeContent()` does not easily give a way to discern between the attribute not being present and the attribute being present, but with no value (i.e. `<item attr=' '/>`).

If you need to be able to discern between the two, use `xmlTreeGetAttributes()` to first see if the attribute is present, and if so, then use `xmlTreeGetContent()` on the attribute.
If multiple attributes with the same local name exist with different namespaces and no ns URI is given, the first one it finds is returned. If you want a specific attribute, provide the proper ns URI for it (which could be '' for the default namespace).

SEE ALSO

xmlTreeGetName, xmlTreeGetContent, xmlTreeGetAllContent, xmlTreeGetLine, xmlTreeGetType, xmlTreeIsBlankNode
1.6.51 xmlTreeGetAttributes

SYNOPSIS

```
xmlNode[] xmlTreeGetAttributes(xmlNode element [, string name [, string ns_URI ]])
```

Parameters:

- `element` - the element to get the attribute(s) from
- `name` (*optional*) - the name of the attribute to get. If not specified, all attributes are returned.
- `ns_URI` (*optional*) - limit result to the attributes in the namespace `ns_URI`. If not specified, namespace is ignored.

Returns:

- the `xmlNode(s)` of the attribute(s) requested

DESCRIPTION

`xmlTreeGetAttributes()` returns the attributes of an element. If a `name` is specified, then only the attribute matching that name is returned (if it exists).

If no `ns_URI` is specified, then the namespace prefix/URIs are disregarded and only local names are compared. If a `ns_URI` is provided, only an attribute that matches a name and the namespace URI will be returned. This includes the empty namespace; if you pass in a `ns_URI` as ‘ ‘, then only the attribute matching that name and without a namespace will be returned.

EXAMPLE

```
<$stockAttr = (xmlTreeGetAttributes($node, 'stock'))>
```

SEE ALSO

`xmlTreeGetFirstAttribute`
1.6.52 xmlTreeGetChildren

SYNOPSIS

```
xmlNode[] xmlTreeGetChildren(xmlNode parent [, string name
[, string ns_URI]])
```

Parameters:

- `element` - the element to get the children from
- `name` (optional) - only get children with the specified local name. If not specified, all children are returned.
- `ns_URI` (optional) - limit result to the children in the namespace `ns_URI`. If not specified, namespace is ignored.

Returns:

- the `xmlNode` children of the element `parent`

DESCRIPTION

`xmlTreeGetChildren()` returns the children of the element `parent`, which includes elements, comments, CDATA, etc.

If you only want certain elements, you can specify a `name` parameter to only get elements named `name`. Additionally, if no `ns_URI` is specified, then the namespace prefix/URIs are disregarded and only local names are compared. If a `ns_URI` is provided, only an attribute that matches a name and the namespace URI will be returned. This includes the empty namespace; if you pass in a `ns_URI` as '', then only the attribute matching that name and without a namespace will be returned.

EXAMPLE

```
<$children = (xmlTreeGetChildren($parent, 'item'))>
```

SEE ALSO

`xmlTreeGetFirstChild, xmlTreeGetNext, xmlTreeGetPrevious, xmlTreeGetParent, xmlTreeGetChildrenContent`
1.6.53 xmlTreeGetChildrenContent

SYNOPSIS

string[] xmlTreeGetChildrenContent(xmlNode parent, string name [, string ns_URI])

Parameters:

- parent - the element to get the children’s content from
- name - the local name of the child elements to get content from
- ns_URI (optional) - limit result to the children in the namespace ns_URI. If not specified, namespace is ignored.

Returns:

- the content of the named children

DESCRIPTION

xmlTreeGetChildrenContent() provides an easy way to read “config” XML data, where a node has many element children. Rather than getting each child element and getting their content, xmlTreeGetChildrenContent() allows you to do it in one call. If there are multiple name child elements, then multiple strings are returned.

The call

<$content = (xmlTreeGetChildrenContent($node, 'item'))>

is essentially a shortcut for the following:

<$tmp = (xmlTreeGetChildren($node, 'item'))>
<loop $tmp>
  <$ret = (xmlTreeGetAllContent($tmp))>
  <$content = $content $ret>
</loop>

See the xmlTree02_Get sample script for examples of using xmlTreeGetChildrenContent().

If no ns_URI is specified, then the namespace prefix/URIs are disregarded and only local names are compared. If a ns_URI is provided, only elements that match the name and the namespace URI will be returned. This includes the empty namespace; if you pass in a ns_URI as ’ ’, then only the attribute matching that name and without a namespace will be returned.
EXAMPLE

```php
<$content = (xmlTreeGetChildrenContent($parent, 'item'))>
```

SEE ALSO

xmlTreeDumpNode, xmlTreeGetAllContent, xmlTreeGetContent,
xmlTreeGetFirstChild, xmlTreeGetNext, xmlTreeGetPrevious, xmlTreeGetParent
CHAPTER 1. TEXIS WEB SCRIPT

1.6.54 xmlTreeGetContent

SYNOPSIS

string xmlTreeGetContent(xmlNode node [, string options])

Parameters:

- node - the node to get the content of
- options - content options (see below)

Returns:

- the content of node

DESCRIPTION

xmlTreeGetContent() gets the content of the node node. If node is an element, then it returns the concatenation of this element’s text nodes.

Currently options only has one possible value:

- NO_INLINE - By default entity references are replaced with their proper values. For example on the following XML document:

    <?xml version="1.0"?>
    <!DOCTYPE rootNode [ 
    <!ELEMENT rootNode (nodeChild)> 
    <!ELEMENT nodeChild (#PCDATA)> 
    <!ENTITY ts "Thunderstone Software, LLC."}> 
    <rootNode> 
    <nodeChild>Is 5&gt;3? Yes it is! ts is '&ts;'</nodeChild> 
    </rootNode>

    By default, calling xmlTreeGetContent() on the element nodeChild (or its text child) would return

    Is 5>3? Yes it is! ts is 'Thunderstone Software, LLC.'

    The entities &gt; and &ts; were substituted. You can disable the substitution by passing

    NO_INLINE as an option. Doing so would give the following text from xmlTreeGetContent():

    Is 5&gt;3? Yes it is! ts is '&ts;'
EXAMPLE

<$content = (xmlTreeGetContent($node))>

SEE ALSO

xmlTreeGetName, xmlTreeGetAllContent, xmlTreeGetChildrenContent,
xmlTreeGetLine, xmlTreeGetType, xmlTreeIsBlankNode,
xmlTreeGetAttributeContent
xmlTreeGetDoc

SYNOPSIS

```c
xmlDoc xmlTreeGetDoc(xmlNode node)
```

Parameters:

- `node` - the `xmlNode` you want the `xmlDoc` of

Returns:

- the `xmlDoc` of `node`

DESCRIPTION

`xmlTreeGetDoc()` fetches the root of the node’s tree, the `xmlDoc`. If called on an `xmlDoc`, then the doc itself is returned.

EXAMPLE

```c
<$doc = (xmlTreeGetDoc($node))>
```

SEE ALSO

`xmlTreeGetRootElement, xmlTreeGetChildren, xmlTreeGetFirstChild, xmlTreeGetNext, xmlTreeGetPrevious, xmlTreeGetParent`
### 1.6.56 xmlTreeGetEncoding

**SYNOPSIS**

```c
string xmlTreeGetEncoding(xmlTree tree)
```

**Parameters:**

- `doc` - the xmlDoc object

**Returns:**

- The encoding of the document.

**DESCRIPTION**

`xmlTreeGetEncoding()` returns the character encoding used in the document, a set by the XML declaration, such as:

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
```

If no encoding is given in a declaration (or no declaration is present), then `xmlTreeGetEncoding` will return nothing (although UTF-8 is assumed as the encoding, as defined in the XML spec).

**EXAMPLE**

```php
$encoding = (xmlTreeGetEncoding($doc))
```

**SEE ALSO**

`xmlTreeGetVersion`
xmlTreeGetEntityType

SYNOPSIS

string xmlTreeGetEntityType(xmlNode entity)

Parameters:

- entity - the XML_ENTITY_DECL you want to get the type from

Returns:

- the entity type of entity

DESCRIPTION

xmlTreeGetEntityType() will tell you what the type of an entity the node entity is. All entity declarations have an xmlNode type of XML_ENTITYDECL (as retrieved with xmlTreeGetType()), but there are multiple types of entities. The possible values, and examples that would return that type, are:

- XML_INTERNAL_GENERAL_ENTITY - This is the normal entity type, parsed and contained within the document.
  - <!ENTITY ts "Thunderstone Software, LLC.">
- XML_EXTERNAL_GENERAL_PARSED_ENTITY - an external entity that gets parsed.
  - <!ENTITY ep SYSTEM "externalParsed.txt">
- XML_INTERNAL_PARAMETER_ENTITY - an entity that is marked to only be used within the DTD.
  - <!ENTITY % p "(#PCDATA)">
- XML_EXTERNAL_PARAMETER_ENTITY - an external entity that is only to be used within the DTD.
  - <!ENTITY % epar SYSTEM "epar.txt">

EXAMPLE

<$dtd = (xmlTreeGetInternalSubset($doc))>
<$node = (xmlTreeGetFirstChild($dtd))>
<$type = (xmlTreeGetType($node))>
<if 'XML_ENTITY_DECL' eq $type>
  <$etype = (xmlTreeGetEntityType($node))>
</if>
1.6. XML API

SEE ALSO

xmlTreeGetName, xmlTreeGetContent
1.6.58 xmlTreeGetExternalID

SYNOPSIS

    string xmlTreeGetExternalID(xmlNode node)

Parameters:

• node - the XML_DTD_NODE or XML_ENTITYDECL you want to get the External ID from

Returns:

• the External ID from the DTD subset / entity node

DESCRIPTION

xmlTreeGetExternalID() gets the PUBLIC External identifier of a DTD or entity.

EXAMPLE

In the following XML document:

```xml
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE top PUBLIC "-//Thunderstone//Simple Test//EN" "testExternal.dtd">
<top>
...
</top>
```

Calling

```php
<$ext = (xmlTreeGetExternalID($dtd))>
```

would return -//Thunderstone//Simple Test//EN

SEE ALSO

xmlTreeGetSystemID
xmlTreeGetExternalSubset

SYNOPSIS

```
xmLNode xmlTreeGetExternalSubset(xmlDoc doc)
```

Parameters:

- `doc` - the doc you want to get the external DTD from

Returns:

- the `xmlNode` for the external DTD subset of `doc`

DESCRIPTION

`xmlTreeGetExternalSubset()` gets the external DTD subset from the XML document `doc`. This is the part of the DTD that is defined via a SYSTEM reference, as opposed to the internal subset, which is defined within the XML file itself.

Note that the XML file must be parsed with the option `XML_PARSE_DTDLOAD`, otherwise the external DTD will not be loaded.

EXAMPLE

```
<$dtd = (xmlTreeGetExternalSubset($doc))>
```

CAVEATS

Parsing the external DTD may involve performing a network fetch. This can be disabled by adding `XML_PARSE_NONET` to the parsing options.

SEE ALSO

`xmlTreeGetInternalSubset`
1.6.60 xmlTreeGetFirstAttribute

SYNOPSIS

xmlNode xmlTreeGetFirstAttribute(xmlNode element)

Parameters:

- element - the element you want to get the first attribute of

Returns:

- the first attribute of element

DESCRIPTION

xmlTreeGetFirstAttribute() returns the first attribute of the element element.

Note that this returns the xmlNode of the attribute, not the text of it. With the xmlNode of the attribute, you can use things like xmlTreeGetName(), xmlTreeGetContent(), xmlTreeGetNext(), etc. If you only want the content of a certain attribute, you can use xmlTreeGetAttributeContent() directly on the element instead.

EXAMPLE

<$attr = (xmlTreeGetFirstAttribute($element))>

SEE ALSO

xmlTreeGetChildren, xmlTreeGetFirstChild, xmlTreeGetNext, xmlTreeGetPrevious, xmlTreeGetParent, xmlTreeGetAttributes
1.6.61 xmlTreeGetFirstChild

SYNOPSIS

xmlNode xmlTreeGetFirstChild(xmlNode node)

Parameters:

- node - the node to get the first child of

Returns:

- the first child of node

DESCRIPTION

xmlTreeGetFirstChild() gets the first child of a node.

Note that the first child of an element might not be its first element child; there could be a comment, CDATA, or a blank text node (see the Text Nodes and Children section, p. 349, for more information).

EXAMPLE

<$child = (xmlTreeGetFirstChild($node))>

SEE ALSO

xmlTreeGetDoc, xmlTreeGetRootElement, xmlTreeGetChildren, xmlTreeGetFirstChild, xmlTreeGetNext, xmlTreeGetPrevious, xmlTreeGetParent
1.6.62  xmlTreeGetInternalSubset

SYNOPSIS

xmlNode xmlTreeGetInternalSubset(xmlDoc doc)

Parameters:

- `doc` - the doc you want to get the internal DTD from

Returns:

- the xmlNode for the internal DTD subset of doc

DESCRIPTION

xmlTreeGetInternalSubset() gets the internal DTD subset from the XML document `doc`. This is the part of the DTD that is defined within the XML file itself, as opposed to an external DTD referenced.

EXAMPLE

```$dtd = (xmlTreeGetInternalSubset($doc))```

SEE ALSO

xmlTreeGetExternalSubset
1.6.63  xmlTreeGetLine

SYNOPSIS

    int xmlTreeGetLine(xmlNode node)

Parameters:

• node - the xmlNode you want the line number of

Returns:

• the line number that node occurs on

DESCRIPTION

xmlTreeGetLine() returns the line number of node. Unlike the reader’s xmlReaderGetLine() function, where the parser may read ahead, the xmlTreeGetLine() function will be accurate.

EXAMPLE

    <$line = (xmlTreeGetLine($node))>

SEE ALSO

xmlTreeGetName, xmlTreeGetContent, xmlTreeGetAllContent, xmlTreeGetType, xmlTreeIsBlankNode, xmlTreeGetAttributeContent
xmlTreeGetName

SYNOPSIS

string xmlTreeGetName(xmlNode node)

Parameters:

• node - the xmlNode you want the name of

Returns:

• the name of node

DESCRIPTION

xmlTreeGetName() gets the name of the node node. For elements and attributes, this is the name of the element or attribute. For things without names, like text nodes, a constant value is returned (text in the case of text nodes).

EXAMPLE

<$name = (xmlTreeGetName($node))>

SEE ALSO

xmlTreeGetName, xmlTreeGetContent, xmlTreeGetAllContent, xmlTreeGetLine, xmlTreeGetType, xmlTreeIsBlankNode, xmlTreeGetAttributeContent
xmlTreeGetNext

SYNOPSIS

xmlNode xmlTreeGetNext(xmlNode item)
xmlNs  xmlTreeGetNext(xmlNs item)

Parameters:

- item - the xmlNode/xmlNs you want the next sibling of

Returns:

- the next sibling of item

DESCRIPTION

xmlTreeGetNext() gets the next sibling of item. This applies for the normal hierarchal tree (elements, comments, CDATA, etc), and things that exist outside the normal tree (attributes and namespaces).

EXAMPLE

<$node = (xmlTreeGetNext($node))>

SEE ALSO

xmlTreeGetChildren, xmlTreeGetFirstChild, xmlTreeGetPrevious, xmlTreeGetParent
1.6.66 xmlTreeGetNs

SYNOPSIS

xmlNs xmlTreeGetNs(xmlNode node)

Parameters:

- node - the element/attribute you want to get the namespace of

Returns:

- the assigned namespace for node

DESCRIPTION

xmlTreeGetNs() gets the assigned namespace for an element or attribute.

Note that this is the namespace that applies to the node, which may be created on a different node, and there may be other namespaces declared on this node that this node isn’t assigned to.

For example, for the XML data

```xml
<top xmlns:ex="http://www.example.com">
    <ex:item xmlns:ts="http://www.thunderstone.com"
             xmlns:web="http://www.webinator.com"/>
</top>
```

calling xmlTreeGetNs() on the <item> node would return the xmlNs for the http://www.example.com namespace, as that is what <item> is assigned to (as opposed to the other two namespaces that are declared on the <item> element).

To get the namespace(s) declared on an element, use xmlTreeGetNsDef().

EXAMPLE

```perl
<$ns = (xmlTreeGetNs($node))>
<$prefix = (xmlTreeGetNsPrefix($ns))>
<$URI = (xmlTreeGetNsURI($ns))>
namespace $URI uses the prefix $prefix
```

SEE ALSO

xmlTreeGetNsDef
**1.6.67 xmlTreeGetNsDef**

**SYNOPSIS**

```
xmlNs[] xmlTreeGetNsDef(xmlNode element)
```

Parameters:

- `element` - the element that you want to get the namespace declarations from

Returns:

- the namespace declarations from `element`

**DESCRIPTION**

`xmlTreeGetNsDef()` gets the namespaces defined on the element `element`. For example, for the following XML data:

```xml
<top xmlns:ex="http://www.example.com">
  <ex:item xmlns:ts="http://www.thunderstone.com"
           xmlns:web="http://www.webinator.com"/>
</top>
```

calling `xmlTreeGetNsDef()` on the `<item>` node would return the `xmlNs` for the `http://www.thunderstone.com` and `http://www.webinator.com` namespaces, as they are defined on `<item>`.

To get the namespace that applies to a node, use `xmlTreeGetNs()`.

**EXAMPLE**

```xml
<$ns = (xmlTreeGetNsDef($node))>
<loop $ns>
  <$prefix = (xmlTreeGetNsPrefix($ns))>
  <$URI = (xmlTreeGetNsURI($ns))>
  namespace $URI uses the prefix $prefix
</loop>
```

**SEE ALSO**

`xmlTreeGetNs`
1.6.68 xmlTreeGetNsURI

SYNOPSIS

    string xmlTreeGetNsURI(xmlNs ns)

Parameters:

• ns - the namespace to get the URI from

Returns:

• the URI of the namespace ns

DESCRIPTION

xmlTreeGetNsURI() returns the URI of a namespace node. The URI can be either a URL
(http://www.thunderstone.com) or a URN (urn:Thunderstone), as defined by RFC 2396.

EXAMPLE

    <$uri = (xmlTreeGetNsURI($ns))>

SEE ALSO

xmlTreeGetNsPrefix, xmlTreeLookupNsURI, xmlTreeLookupNsPrefix
1.6.69  xmlTreeGetNsPrefix

SYNOPSIS

    string xmlTreeGetNsPrefix(xmlNs ns)

Parameters:

    • ns - the namespace to get the prefix of

Returns:

    • the prefix of the namespace ns

DESCRIPTION

xmlTreeGetNsPrefix() gets the prefix of a namespace.

EXAMPLE

    <$prefix = (xmlTreeGetNsPrefix($ns))>

SEE ALSO

xmlTreeGetNsURI, xmlTreeLookupNsURI, xmlTreeLookupNsPrefix
1.6.70  xmlTreeGetParent

SYNOPSIS

    xmlNode xmlTreeGetParent(xmlNode node)

Parameters:

- node - the xmlNode to get the parent of

Returns:

- the parent of node

DESCRIPTION

xmlTreeGetParent() returns the parent of the node node. If this is used on one of the top-level nodes (such as the root element or a comment outside of the XML tree), then the xmlDoc will be returned.

EXAMPLE

    <$parent = (xmlTreeGetParent($node))>

SEE ALSO

xmlTreeGetChildren, xmlTreeGetFirstChild, xmlTreeGetNext, xmlTreeGetPrevious
1.6.71 xmlTreeGetPrevious

SYNOPSIS

```c
xmlNode xmlTreeGetPrevious(xmlNode node)
```

Parameters:

- `node` - the node to get the previous sibling of

Returns:

- the previous sibling of `node`

DESCRIPTION

`xmlTreeGetPrevious()` gets the previous sibling of `node`. It is the opposite of `xmlTreeGetNext()`.

EXAMPLE

```c
<$prev = (xmlTreeGetPrevious($node))>
```

SEE ALSO

`xmlTreeGetChildren`, `xmlTreeGetFirstChild`, `xmlTreeGetNext`, `xmlTreeGetParent`
1.6.72 xmlTreeGetRootElement

SYNOPSIS

xmlNode xmlTreeGetRootElement(xmlDoc doc)

Parameters:

- doc - the document you want to get the root element from

Returns:

- the root element of the doc

DESCRIPTION

xmlTreeGetRootElement() gets the root element from the XML document doc. This is the proper way to get the root element (as opposed to xmlTreeGetFirstChild() on the doc), as described in the xmlDoc vs Root Element (p. 350) section.

EXAMPLE

<$root = (xmlTreeGetRootElement($doc))>

SEE ALSO

xmlTreeGetDoc, xmlTreeGetChildren, xmlTreeGetFirstChild, xmlTreeGetNext, xmlTreeGetPrevious, xmlTreeGetParent
1.6. XML API

1.6.73 xmlTreeGetSystemID

SYNOPSIS

string xmlTreeGetSystemID(xmlNode node)

Parameters:

- node - the XML_DTD_NODE or XML_ENTITY_DECL you want to get the System ID from

Returns:

- the System ID from the dtd subset node

DESCRIPTION

xmlTreeGetSystemID() gets SYSTEM identifier of an external DTD or entity. This is either a local file, or a web address.

EXAMPLE

In the following XML document:

```xml
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE top PUBLIC "-//Thunderstone//Simple Test//EN" "testExternal.dtd">
<top>
...
</top>
```

Calling

```php
<$ext = (xmlTreeGetSystemID($dtd))>
```

would return testExternal.dtd

SEE ALSO

xmlTreeGetSystemID
1.6.74 xmlTreeGetType

SYNOPSIS

    string xmlTreeGetType(xmlNode node)

Parameters:

    • node - the xmlNode to get the type of

Returns:

    • the type of the node

DESCRIPTION

xmlTreeGetType() returns the type of the xmlNode, which tells you whether it’s an element, attribute, comment, etc. The possible returned values are:

    • XML_ELEMENT_NODE
    • XML_ATTRIBUTE_NODE
    • XML_TEXT_NODE
    • XML_CDATA_SECTION_NODE
    • XML_ENTITY_REF_NODE
    • XML_ENTITY_NODE
    • XML_PI_NODE
    • XML_COMMENT_NODE
    • XML_DOCUMENT_NODE
    • XML_DOCUMENT_TYPE_NODE
    • XML_DOCUMENT_FRAG_NODE
    • XML_NOTATION_NODE
    • XML_HTML_DOCUMENT_NODE
    • XML_DTD_NODE
    • XML_ELEMENT_DECL
1.6. XML API

- XML_ATTRIBUTE_DECL
- XML_ENTITY_DECL
- XML_NAMESPACE_DECL
- XML_XINCLUDE_START
- XML_XINCLUDE_END
- XML_DOGB_DOCUMENT_NODE

EXAMPLE

<$type = (xmlTreeGetType($node))>

SEE ALSO

xmlTreeGetName, xmlTreeGetContent, xmlTreeGetAllContent, xmlTreeGetLine, xmlTreeIsBlankNode, xmlTreeGetAttributeContent
xmlTreeGetVersion

SYNOPSIS

```c
string xmlTreeGetVersion(xmlDoc doc)
```

Parameters:

- `doc` - the `xmlDoc` to get the XML version from

Returns:

- the XML version of the data

DESCRIPTION

`xmlTreeGetVersion()` returns the XML version used by the XML data, as defined in the XML declaration:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<top>
  ...
</top>
```

For this data, `xmlTreeGetVersion()` would return `1.0`.

EXAMPLE

```c
<$version = (xmlTreeGetVersion($doc))>
```

SEE ALSO

`xmlTreeGetEncoding`
1.6.76 xmlTreeIsBlankNode

SYNOPSIS

    int xmlTreeIsBlankNode(xmlNode node)

Parameters:

    - node - the xmlNode in question

Returns:

    - 0 - the node is not a blank text node
    - 1 - the node is a blank text node

DESCRIPTION

xmlTreeIsBlankNode() tells you whether the node is a blank text node. The only time this returns 1 is if node is either a text or a CDATA node, and only contains whitespace (as defined by the XML spec), which consists of:

    - space (0x20)
    - horizontal tab (0x09)
    - newlines (0x0A or 0x0D)

EXAMPLE

    <$ret = (xmlTreeIsBlankNode($node))>

SEE ALSO

xmlTreeGetName, xmlTreeGetContent, xmlTreeGetAllContent, xmlTreeGetLine, xmlTreeGetType, xmlTreeGetAttributeContent
1.6.77 xmlTreeLookupNsURI

SYNOPSIS

xmlNs xmlTreeLookupNsURI(xmlNode node, string URI)

Parameters:

- node - the node you want to look up the namespace from
- URI - the namespace URI you want to look up

Returns:

- the xmlNs for the URI namespace.

DESCRIPTION

xmlTreeLookupNsURI() looks up a namespace based on the URI, if it exists. Handy if you want to assign a namespace to a node and you need to know if it’s already declared in this scope so you don’t create an unnecessary duplicate.

This does not search the entire XML tree, but only the namespace scope of the given node, which is the current node and all nodes above it. If a namespace isn’t available for a node to use, then it won’t be found from a search on that node.

For example, given the following document:

```
<rootNode>
    <container xmlns:foo="http://www.example.com">
        <item1 xmlns:bar="http://www.example.com">text</item1>
        <item2>text</item2>
    </container>
    <item3>text</item3>
</rootNode>
```

If you look up the http://www.example.com prefix from:

- <item1>, we’ll get bar
- <item2>, we’ll get foo
- <item3>, we’ll get nothing, because the namespace http://www.example.com is not defined for <item3>’s scope.
EXAMPLE

<$ns = (xmlTreeLookupNsURI($node, 'http://www.example.com'))>

SEE ALSO

xmlTreeGetNsURI, xmlTreeGetNsPrefix, xmlTreeLookupNsPrefix
1.6.78 xmlTreeLookupNsPrefix

SYNOPSIS

xmlNs xmlTreeLookupNsPrefix(xmlNode node, string prefix)

Parameters:

• node - the xmlNode to lookup the namespace from
• prefix - the prefix of the namespace to lookup

Returns:

• the namespace that uses prefix

DESCRIPTION

xmlTreeLookupNsPrefix() looks up the namespace in the scope of node that uses the prefix prefix. Useful if you know that a namespace exists in the tree but you don’t have the xmlNs of it.

This does not search the entire XML tree, but only the namespace scope of the given node, which is the current node and all nodes above it. If a namespace isn’t available for a node to use, then it won’t be found from a search on that node.

For example, given the following document:

```xml
<rootNode>
  <container xmlns:foo="http://www.example.com/container">
    <item1 xmlns:foo="http://www.example.com/item">text</item1>
    <item2>text</item2>
  </container>
  <item3>text</item3>
</rootNode>
```

If you look up the foo prefix from:

• <item1>, we’ll get http://www.example.com/item
• <item2>, we’ll get http://www.example.com/container
• <item3>, we’ll get nothing, because the prefix foo is not defined for <item3>’s scope.

EXAMPLE

```php
<$ns = (xmlTreeLookupNsPrefix($node, 'ts'))>
```
SEE ALSO

xmlTreeGetNsURI, xmlTreeGetNsPrefix, xmlTreeLookupNsURI
xmlTreeNewAttribute

SYNOPSIS

xmlNode xmlTreeNewAttribute(xmlNode parent, string name, [, string content[, xmlNs ns]])

Parameters:

- **parent** - the xmlNode to create the attribute on
- **name** - the name of the new attribute
- **content (optional)** - the content for the attribute
- **ns (optional)** - the namespace to use for the attribute

Returns:

- the xmlNode for the new attribute

DESCRIPTION

xmlTreeNewAttribute() creates a new attribute. Attribute names must be unique, so if another attribute with the same name already exists, the existing one will be unlinked.

EXAMPLE

```<$attr = (xmlTreeNewAttribute($element, 'id', '12345'))>
```

SEE ALSO

xmlTreeNewElement, xmlTreeNewPI, xmlTreeNewText, xmlTreeNewCDATA, xmlTreeNewComment, xmlTreeNewNs
1.6.80  xmlTreeNewCDATA

SYNOPSIS

    xmlNode xmlTreeNewCDATA(xmlNode parent, string content)

Parameters:

- **parent** - the parent of the new CDATA section (can specify none with '')
- **content** - the content for the new CDATA section

Returns:

- the xmlNode for the new CDATA section

DESCRIPTION

xmlTreeNewCDATA() creates a new CDATA block, which is a block of text that goes unparsed by the XML parser. See http://www.w3.org/TR/2006/REC-xml-20060816/#sec-cdata-sect for more information on CDATA sections.

EXAMPLE

```xml
<$cdata = (xmlTreeNewCDATA($node,
    'I go unparsed! <<<<< Nya nya!'))>
```

Will produce the following in the output:

```xml
<![CDATA[I go unparsed! <<<<< Nya nya!]]>
```

SEE ALSO

xmlTreeNewElement, xmlTreeNewPI, xmlTreeNewText, xmlTreeNewComment
xmlTreeNewComment

**SYNOPSIS**

```c
xmlNode xmlTreeNewComment(xmlNode parent, string content)
```

**Parameters:**

- `parent` - the parent of the new comment node (can specify none with '')
- `content` - the content of the new comment

**Returns:**

- the `xmlNode` of the new comment

**DESCRIPTION**

`xmlTreeNewComment()` creates a new comment as a child of `parent`.

**EXAMPLE**

```c
<$comment = (xmlTreeNewComment($parent, 'important!'))>
```

Will produce the following in output:

```
<!-- important! -->
```

**SEE ALSO**

`xmlTreeNewElement`, `xmlTreeNewPI`, `xmlTreeNewText`, `xmlTreeNewCDATA`
1.6. XML API

1.6.82 xmlTreeNewDoc

SYNOPSIS

```c
xmlDoc xmlTreeNewDoc(string version)
```

Parameters:

- `version` - the XML version for the doc (usually 1.0 or 1.1)

Returns:

- the new `xmlDoc`

DESCRIPTION

`xmlTreeNewDoc()` is for creating new XML documents from scratch. You must provide a version number, which becomes the version in the XML prolog.

EXAMPLE

```c
$doc = (xmlTreeNewDoc('1.0'))
```

Will create a new XML document that will have the following XML prolog: `<?xml version="1.0"?>`

CAVEATS

While the XML version is determined at document-creation time, the encoding, which also goes in the XML declaration, is determined by parameters when the document is saved/printed. See `xmlTreePrintDoc()` and `xmlTreeSaveDoc()` for more information.

SEE ALSO

`xmlTreeNewDocFromFile`, `xmlTreeNewDocFromString`, `xmlTreePrintDoc`, `xmlTreeSaveDoc`
xmlTreeNewDocFromFile

SYNOPSIS

xmlDoc xmlTreeNewDocFromFile(string filename [, string encoding] [, string options])

Parameters:

- **filename** - the filename of the XML file to read
- **encoding** *optional* - encoding to use for the file. If none given, the parser will attempt to discover on its own.
- **options** *optional* - a comma-separated list of parsing options (see below)

Returns:

- the parsed xmlDoc of the file

DESCRIPTION

xmlTreeNewDocFromFile() reads the XML data from the file filename and construct an xmlDoc for it.

The possible values for the options parameter are:

- XML_PARSE_RECOVER - recover on errors
- XML_PARSE_NOENT - substitute entities
- XML_PARSE_DTDLOAD - load the external subset
- XML_PARSE_DTDATTR - default DTD attributes
- XML_PARSEDTDVALID - validate with the DTD
- XML_PARSE_NOERROR - suppress error reports
- XML_PARSE_NOWARNING - suppress warning reports
- XML_PARSE_PEDANTIC - pedantic error reporting
- XML_PARSE_NOBLANKS - remove blank nodes
- XML_PARSE_XINCLUDE - Implement XInclude substitution
- XML_PARSE_NONET - Forbid network access
1.6. XML API

- XML_PARSE_NSIdentity - remove redundant namespaces declarations
- XML_PARSE_NOCDATA - merge CDATA as text nodes
- XML_PARSE_NOXINCNODE - do not generate XINCLUDE START/END nodes

EXAMPLE

```xml
<$doc = (xmlTreeNewDocFromFile($filename, 'XML_PARSE_NOBLANKS'))>
```

SEE ALSO

xmlTreeNewDoc, xmlTreeNewDocFromFile, xmlTreePrintDoc, xmlTreeSaveDoc
xmlTreeNewDocFromString

SYNOPSIS

xmlDoc xmlTreeNewDocFromString(string data [, string encoding] [, string options])

Parameters:

- data - the XML data to parse
- encoding optional - encoding to use for the file. If none given, the parser will attempt to discover on its own.
- options optional - a comma-separated list of parsing options (see below)

Returns:

- the parsed xmlDoc from the text data

DESCRIPTION

xmlTreeNewDocFromString() parses the XML data contained within the string data and returns the xmlDoc for it.

The possible values for the options parameter are:

- XML_PARSE_RECOVER - recover on errors
- XML_PARSE_NOENT - substitute entities
- XML_PARSE_DTDLOAD - load the external subset
- XML_PARSE_DTDATTR - default DTD attributes
- XML_PARSE/DTDVALID - validate with the DTD
- XML_PARSE_NOERROR - suppress error reports
- XML_PARSE_NOWARNING - suppress warning reports
- XML_PARSE_PEDANTIC - pedantic error reporting
- XML_PARSE_NOBLANKS - remove blank nodes
- XML_PARSE_XINCLUDE - Implement XInclude substitution
- XML_PARSE_NONET - Forbid network access
1.6. XML API

- `XML_PARSE_NSCLEAN` - remove redundant namespaces declarations
- `XML_PARSE_NOCDATA` - merge CDATA as text nodes
- `XML_PARSE_NOXINCNODE` - do not generate XINCLUDE START/END nodes

EXAMPLE

```xml
<capture>
  <?xml version="1.0"?>
  <top>
    <item>I'm an item!</item>
  </top>
</capture><$xml = $ret>
<$doc = (xmlTreeNewDocFromString($xml, 'XML_PARSE_NOBLANKS'))>

SEE ALSO

xmlTreeNewDoc, xmlTreeNewDocFromFile, xmlTreePrintDoc, xmlTreeSaveDoc
1.6.85 xmlTreeNewElement

SYNOPSIS

xmlNode xmlTreeNewElement(xmlNode parent, string name,
    [, string content[, xmlNs ns]])

Parameters:

• parent - the xmlNode to create the element on
• name - the name of the new element
• content (optional) - the text content for the element
• ns (optional) - the namespace to use for the element

Returns:

• the xmlNode for the new element

DESCRIPTION

xmlTreeNewElement() creates a new element as a child of parent. If the parent is an element in an xmlns namespace, the child will automatically be created in the same namespace.

EXAMPLE

<$elem1 = (xmlTreeNewElement($root, 'item'))>

SEE ALSO

xmlTreeNewPI, xmlTreeNewText, xmlTreeNewCDATA, xmlTreeNewComment
1.6.86 xmlTreeNewNs

SYNOPSIS

xmlNs xmlTreeNewNs(xmlNode element, string prefix, string URI)

Parameters:

- parent - the element that will be parent of this xmlNs
- prefix - the prefix for the new namespace. ' ' may be used to indicate no prefix.
- URI - the URI for the new namespace. ' ' may be used for the default, empty namespace.

Returns:

- the xmlNs of the new namespace

DESCRIPTION

xmlTreeNewNs() creates a new namespace declaration on element. This xmlNs can then be assigned to nodes via xmlTreeSetNs() to put nodes in that namespace.

EXAMPLE

Given an XML document

```xml
<rootNode>
  <item>stuff</item>
</rootNode>
```

calling the following

```php
<$root = (xmlTreeGetRootElement($doc))>
<$ns = (xmlTreeNewNs($root, 'ts', 'http://www.thunderstone.com'))>
<$ret = (xmlTreeSetNs($root, $ns))>
```

and printing the doc will produce

```xml
  <item>stuff</item>
</ts:rootNode>
```
SEE ALSO

xmlTreeNewElement, xmlTreeNewPI, xmlTreeNewText, xmlTreeNewCDATA,
xmlTreeNewComment, xmlTreeNewAttribute
1.6. XML API

1.6.87 xmlTreeNewPI

SYNOPSIS

xmlNode xmlTreeNewPI(xmlNode parent, string target,
string content)

Parameters:

- parent - the xmlNode that will be the parent of the new processing instruction
- target - the application to which the processing instruction is directed
- content - the content for the new processing instruction

Returns:

- the xmlNode for the new processing instruction

DESCRIPTION

xmlTreeNewPI() creates a new processing instruction as a child of parent. Processing instructions are commonly used to assign a stylesheet to XML data. See http://www.w3.org/TR/REC-xml/#sec-pi for more information.

EXAMPLE

```xml
<$pi = (xmlTreeNewPI($doc, 'xml-stylesheet',
    'type="text/xsl" href="default.xsl"'))>
```

will produce this in the XML output:

```xml
<?xml-stylesheet type="text/xsl" href="default.xsl"?>
```

SEE ALSO

xmlTreeNewElement, xmlTreeNewText, xmlTreeNewCDATA, xmlTreeNewComment
1.6.88 xmlTreeNewText

SYNOPSIS

xmlNode xmlTreeNewText(xmlNode parent, string content)

Parameters:

• parent - the xmlNode that will be the parent of the new text node
• content - the text content of the new text node

Returns:

• the xmlNode for the new text node

DESCRIPTION

xmlTreeNewText() creates a new text node as a child of parent.

This level of control is usually not required for adding text to the XML data. It usually is recommended to use xmlTreeAddContent() or xmlTreeSetContent() instead.

EXAMPLE

<$text = (xmlTreeNewText($parent, 'This is a text node.'))>

SEE ALSO

xmlTreeNewElement, xmlTreeNewPI, xmlTreeNewCDATA, xmlTreeNewComment
1.6.89  xmlTreeNewXPath

SYNOPSIS

    xmlXPath xmlTreeNewXPath(xmlDoc doc)

Parameters:

- doc - the xmlDoc to create the XPath for

Returns:

- the new xmlXPath engine

DESCRIPTION

xmlTreeNewXPath() creates a new xmlXPath engine to use on the XML document doc. The
xmlXPath object is tied to the doc, so each document needs its own xmlXPath object.
This object can then be used in xmlTreeXPathExecute() to execute XPath expressions.

EXAMPLE

    <$xpath = (xmlTreeNewXPath($doc))>

SEE ALSO

xmlTreeXPathExecute, xmlTreeXPathRegisterNs xmlTreeXPathSetContext
1.6.90 xmlTreePrintDoc

SYNOPSIS

string xmlTreePrintDoc(xmlDoc doc [, string encoding]
                      [, string options])

Parameters:

• doc - the xmlDoc to be printed

• encoding - the character encoding to use when serializing the document. Defaults to UTF-8.

• options - the options to use when saving (see below)

Returns:

• the serialized string of doc

DESCRIPTION

xmlTreePrintDoc() returns the serialized, text version of the XML document doc.

The possible values for options are:

• INDENT - add extra whitespace text nodes to indent the output.
<$doc = (xmlTreeNewDoc( '1.0'))>
<$root = (xmlTreeNewElement($doc, 'top'))>
<$ret = (xmlTreeNewElement($root, 'item', 'Look at me!'))>
<$ret = (xmlTreeNewElement($root, 'item', 'Items ahoy!'))>
<$nested = (xmlTreeNewElement($root, 'nested'))>
<$ret = (xmlTreeNewElement($nested, 'item', 'Going deeper!'))>
<$output = (xmlTreePrintDoc($doc))>

default (no indent):
-------------
<fmt "%js" $output>
-------------

<$output = (xmlTreePrintDoc($doc, 'INDENT'))>
with indents:
-------------
<fmt "%js" $output>
-------------

Will produce:

default (no indent):
-------------
<?xml version="1.0"?>
<top><item>Look at me!</item><item>Items ahoy!</item>
<nested><item>Going deeper!</item></nested></top>
-------------

with indents:
-------------
<?xml version="1.0"?>
<top>
  <item>Look at me!</item>
  <item>Items ahoy!</item>
  <nested>
    <item>Going deeper!</item>
  </nested>
</top>
-------------

Note that the "no indent" version isn’t actually two lines, but only appears to due to line length wrapping in the documentation.

SEE ALSO
xmlTreeSaveDoc
1.6.91 xmlTreeQuickXPath

SYNOPSIS

```
string[] xmlTreeQuickXPath(string xmlRaw, string xpathQuery [], string[] xmlns)
```

Parameters:

- `xmlRaw` - the plain text of the XML document you want to extract information from
- `xpathQuery` - the XPath expression that identifies the nodes you want to extract the data from
- `xmlns (optional)` - an array of `prefix=URI` namespaces to use in the XPath query

Returns:

- String values of the node from the XML document `xmlRaw` that match `xpathQuery`

DESCRIPTION

`xmlTreeQuickXPath` allows you to easily extract information from an unparsed XML document in a one-shot function. It is intended to be used in SQL statements to extract specific information from a field that contains XML data.

It is essentially a one statement version of the following:

```
<$doc = (xmlTreeNewDocFromString($xmlRaw))>
<$xpath = (xmlTreeNewXPath($doc))>
<$nodes = (xmlTreeXPathExecute($xpathQuery))>
<loop $nodes>
  <$ret = (xmlTreeGetAllContent($nodes))>
  <$content = $content $ret>
</loop>
```

EXAMPLE

if the `xmlData` field of a table has content like this:

```
<extraInfo>
  <price>8.99</price>
  <author>John Doe</author>
  <isbn>978-0-06-051280-4</isbn>
</extraInfo>
```
Then the following SQL statement will match that row:

```sql
SELECT * from myTable where xmlTreeQuickXPath(data, '/extraInfo/author') = 'John Doe'
```

SEE ALSO

xmlTreeNewXPath, xmlTreeXPathExecute, xmlTreeXPathRegisterNs
1.6.92 xmlTreeSaveDoc

SYNOPSIS

    int xmlTreeSaveDoc(xmlDoc doc, string filename,
                      [, string encoding] [, string options])

Parameters:

- doc - the xmlDoc to be saved to a file.
- filename - the file to save doc to
- encoding *(optional)* - the character encoding to use. Default is UTF-8.
- options *(optional)* - the options to use when saving (see below)

Returns:

- the number of bytes written, or -1 in the case of error.

DESCRIPTION

xmlTreeSaveDoc() writes the XML document doc to the file filename.

The possible values for options are:

- INDENT - add extra whitespace text nodes to indent the output. See xmlTreePrintDoc() for examples.

EXAMPLE

    <$numBytes = (xmlTreeSaveDoc($doc, 'output.xml'))>

SEE ALSO

xmlTreePrintDoc
1.6.93 xmlTreeSetContent

SYNOPSIS

```c
xmlNode xmlTreeSetContent(xmlNode node, string content)
```

Parameters:

- `node` - the `xmlNode` to set the content for
- `content` - the text to set as the content of `node`

Returns:

- the node

DESCRIPTION

`xmlTreeSetContent()` sets the content of `node` to the string `content`.

Note that this replaces all text AND children of `node` with `content`. If you only want to replace the text, you’ll need to browse to the element’s text child and call `xmlTreeSetContent()` on that instead of on the element itself.

EXAMPLE

```c
<$ret = (xmlTreeSetContent($node, 'new content'))>
```

SEE ALSO

`xmlTreeSetName`, `xmlTreeAddContent`
1.6.94 xmlTreeSetName

SYNOPSIS

```c
xmlNode xmlTreeSetName(xmlNode node, string name)
```

Parameters:

- `node` - the `xmlNode` you want to set the name of
- `name` - the name you want to use for `node`

Returns:

- the `node`

DESCRIPTION

`xmlTreeSetName()` sets the name of `node`. Used for setting the name of elements or attributes.

EXAMPLE

```c
<$ret = (xmlTreeSetName($node, 'item'))>
```

SEE ALSO

`xmlTreeSetContent, xmlTreeAddContent`
1.6.95  xmlTreeSetNs

SYNOPSIS

xmlNode xmlTreeSetNs(xmlNode node, xmlNs ns)

Parameters:

- node - the xmlNode to set the namespace for
- ns - the namespace to use for node

Returns:

- the node

DESCRIPTION

xmlTreeSetNs() assigns the namespace ns to be used with node. Note that this is different from xmlTreeNewNs(), which simply declares the namespace to exist, but does not place any nodes in that namespace.

EXAMPLE

Given the simple one-element document <top/>, calling

<$ns = (xmlTreeNewNs($top, 'ts', 'urn:Thunderstone'))>

Would result in the following document: <top xmlns:ts="urn:Thunderstone"/>
The urn:Thunderstone namespace exists, but isn’t being used by <top/>. Additionally calling

<$ret = (xmlTreeSetNs($top, $ns))>

would result in the following document: <ts:top xmlns:ts="urn:Thunderstone"/>
Here the namespace has been declared, and the node is set to it.

CAVEATS

You may have noticed that if using the empty prefix, the xmlTreeSetNs() isn’t necessary for correct output. Making a new ns with no prefix but not setting with xmlTreeSetNs() would produce:
<top xmlns="urn:Thunderstone"/>

While this will result in correct output, it’s still recommended to call xmlTreeSetNs() to assign the namespace to any node that needs it. If the prefix for the urn:Thunderstone ever changed, the prefix on <top> would only be properly set if it was assigned to the namespace with xmlTreeSetNs().

SEE ALSO
xmlTreeGetNs, xmlTreeNewNs
1.6.96  xmlTreeSetNsPrefix

SYNOPSIS

    string xmlTreeSetNsPrefix(xmlNs ns, string prefix)

Parameters:

  • ns - the xmlNs whose prefix you want to change
  • prefix - the new namespace prefix to use for ns

Returns:

  • the old namespace prefix for ns (if any)

DESCRIPTION

xmlTreeSetNsPrefix() changes the namespace prefix for an already-created namespace. Creating a namespace is done with xmlTreeNewNs().

EXAMPLE

    <$oldPrefix = (xmlTreeSetNsPrefix($ns, 'myns'))>

SEE ALSO

xmlTreeSetNsURI, xmlTreeClearNs
1.6.97 xmlTreeSetNsURI

SYNOPSIS

    string xmlTreeSetNsURI(xmlNs ns, string URI)

Parameters:

- `ns` - the xmlNs whose prefix you want to change
- `URI` - the new namespace URI to use for `ns`

Returns:

- the old namespace URI for `ns` (if any)

DESCRIPTION

xmlTreeSetNsURI() changes the namespace URI for an already-created namespace. Creating a namespace is done with xmlTreeNewNs().

EXAMPLE

    <$oldPrefix = (xmlTreeSetNsPrefix($ns, 'urn:Thunderstone'))>

SEE ALSO

xmlTreeClearNs, xmlTreeSetNsURI
1.6.98 xmlTreeSetRootElement

SYNOPSIS

```c
xmlNode xmlTreeSetRootElement(xmlDoc doc, xmlNode element)
```

Parameters:

- `doc` - the `xmlDoc` to set the root element of
- `element` - the element to set as the root of `doc`

Returns:

- the old root element (if any)

DESCRIPTION

`xmlTreeSetRootElement()` sets the root element of `doc` to `element`. If there was already a root element, it is removed and returned.

EXAMPLE

```xml
<$ret = (xmlTreeSetRootElement($doc, $node))>
```

SEE ALSO

`xmlTreeAddChild`, `xmlTreeAddChildList`, `xmlTreeAddNextSibling`, `xmlTreeAddPrevSibling`, `xmlTreeAddSibling`
1.6.99 xmlTreeUnlinkNode

SYNOPSIS

xmlNode xmlTreeUnlinkNode(xmlNode node)

Parameters:

• node - the xmlNode you want to unlink

Returns:

• the node

DESCRIPTION

xmlTreeUnlinkNode() removes the node from the XML document in which it exists. All of its
children will also be removed from the document (but still attached to node).

EXAMPLE

<$ret = (xmlTreeUnlinkNode($node))

SEE ALSO

xmlTreeAddChild, xmlTreeAddChildList, xmlTreeAddNextSibling,
xmlTreeAddPrevSibling, xmlTreeAddSibling, xmlTreeSetRootElement
1.6.100 xmlTreeXPathExecute

SYNOPSIS

xmlNode xmlTreeXPathExecute(xmlXPath xpath, string expression)

Parameters:

- xpath - an xmlXPath object
- expression - the XPath expression to execute

Returns:

- the xmlNode's that match the XPath expression

DESCRIPTION

xmlTreeXPathExecute() executes an XPath expression, returning all xmlNode's that match the expression. It operates on an xmlXPath object, which can be obtained from a doc with xmlTreeNewXPath().

If the specified XPath returns a basic data type such as a string (like name(/ *)) or a number (like string-length(name(/ *))), then it will be contained in a single XML_TEXT_NODE.

EXAMPLE

This example finds all <item>s with a price attribute that's greater than 5, and prints out their <name> sub-element and price attribute.

```xml
<$xpath = (xmlTreeNewXPath($doc))>
<$nodes = (xmlTreeXPathExecute($xpath, '//item[@price>5]'))>
<loop $nodes>
  <$name = (xmlTreeGetChildrenContent($nodes, 'name'))>
  <$price = (xmlTreeGetAttributeContent($nodes, 'price'))>
  $name costs $price
</loop>
```

CAVEATS

In the unusual case that you're XPathing for a namespace declaration directly

(//*/namespace::myprefix), note that the xmlNs object returned is a copy, and changes made to it
will not be reflected in the original doc (elements and attributes return a reference in the original doc, and do not have this limitation).

Instead, you can XPath for the element that contains that definition (//*[namespace::myprefix]), use xmlTreeGetNsDef to get the namespace object and change it as needed.

SEE ALSO

xmlTreeNewXPath, xmlTreeXPathRegisterNs, xmlTreeXPathSetContext
### xmlTreeXPathRegisterNs

#### SYNOPSIS

```c
int xmlTreeXPathRegisterNs(xmlXPath xpath, string prefix, string URI);
```

#### Parameters:

- `xpath` - the xmlXPath to register a namespace for
- `prefix` - the prefix for the namespace
- `URI` - the URI for the namespace

#### Returns:

- 0 - success
- -1 - error

#### DESCRIPTION

xmlTreeXPathRegisterNs() registers a prefix that can be used in an XPath expression. This does not affect the document the xpath is operating on in any way, and only applies to XPath expressions that execute on the xmlXPath object.

#### EXAMPLE

```c
<$xpath = (xmlTreeNewXPath($doc))>
<$ret = (xmlTreeXPathRegisterNs($xpath, 'ts', 'urn:Thunderstone'))>
<$nodes = (xmlTreeXPathExecute($xpath, '//ts:item[@price>5]'))>
```

#### SEE ALSO

xmlTreeNewXPath, xmlTreeXPathExecute
1.6.102 xmlTreeXPathSetContext

SYNOPSIS

xmlNode xmlTreeXPathSetContext(xmlXPath xpath, xmlNode node)

Parameters:

- xpath - the xmlXPath to set a context for
- node - the new starting context for xpath

Returns:

- the previous context

DESCRIPTION

xmlTreeXPathSetContext() lets you set a starting context for the xpath. This lets you perform relative XPath queries, starting from that node, instead of the document root.

Absolute queries (that start with '/') are not affected by the context. The context is the xmlDoc by default, and can be reset by passing the xmlDoc to xmlTreeXPathSetContext.

EXAMPLE

```xml
<capture>
  <?xml version="1.0"?>
  <results>
  <result>
    <data type="name">Result 1</data>
    <data type="location">http://www.example.com/one</data>
  </result>
  <result>
    <data type="name">Result 2</data>
    <data type="location">http://www.example.com/two</data>
  </result>
  </results>
</capture>
$xml = $ret
<doc = (xmlTreeNewDocFromString($xml, 'XML_PARSE_NOBLANKS'))>
<root = (xmlTreeGetRootElement($doc))>
$firstResult = (xmlTreeGetFirstChild($root))>
(xpath = (xmlTreeNewXPath($doc))>
```
<$ret = (xmlTreeXPathSetContext($xpath, $firstResult))>
<capture>data[@type='name']</capture>
<$nodes = (xmlTreeXPathExecute($xpath, $ret))>
<loop $nodes>
  <$ret = (xmlTreeGetContent($nodes))>
  <fmt "Matched %s\n" $ret>
</loop>

SEE ALSO

xmlTreeNewXPath, xmlTreeXPathExecute
1.6.103 xmlWriterEndAttribute

SYNOPSIS

    int xmlWriterEndAttribute(xmlWriter writer)

Parameters:

  • writer - the xmlWriter

Returns:

  • the bytes written (may be 0 because of buffering) or -1 in case of error

DESCRIPTION

xmlWriterEndAttribute() ends the current attribute being written.

EXAMPLE

    <$ret = (xmlWriterEndAttribute($writer))>

SEE ALSO

xmlWriterStartAttribute, xmlWriterWrite, xmlWriterWriteAttribute
1.6.104 xmlWriterEndCDATA

SYNOPSIS

    int xmlWriterEndCDATA(xmlWriter writer)

Parameters:

- writer - the xmlWriter

Returns:

- the bytes written (may be 0 because of buffering) or -1 in case of error

DESCRIPTION

xmlWriterEndCDATA() ends the current CDATA node being written.

EXAMPLE

    <$ret = (xmlWriterEndCDATA($writer))>

SEE ALSO

xmlWriterStartCDATA, xmlWriterWrite, xmlWriterWriteCDATA
1.6.105 xmlWriterEndComment

SYNOPSIS

    int xmlWriterEndComment(xmlWriter writer)

Parameters:

• writer - the xmlWriter

Returns:

• the bytes written (may be 0 because of buffering) or -1 in case of error

DESCRIPTION

xmlWriterEndComment() ends the current comment being written.

EXAMPLE

<$ret = (xmlWriterEndComment($writer))>

SEE ALSO

xmlWriterStartComment, xmlWriterWrite, xmlWriterWriteComment
1.6.106 xmlWriterEndDocument

SYNOPSIS

    int xmlWriterEndDocument(xmlWriter writer)

Parameters:

- writer - the xmlWriter

Returns:

- the bytes written (may be 0 because of buffering) or -1 in case of error

DESCRIPTION

xmlWriterEndDocument() completes the XML document being written. If there are any open elements, they are closed automatically. If this writer was created with xmlWriterNewToFile, then the file handle for the writer is closed.

EXAMPLE

    <$ret = (xmlWriterEndDocument($writer))>

SEE ALSO

xmlWriterGetContent
xmlWriterEndElement

SYNOPSIS

    int xmlWriterEndElement(xmlWriter writer)

Parameters:

  • writer - the xmlWriter

Returns:

  • the bytes written (may be 0 because of buffering) or -1 in case of error

DESCRIPTION

xmlWriterEndElement() ends the current element being written.

EXAMPLE

    <$ret = (xmlWriterEndElement($writer))>

SEE ALSO

xmlWriterStartElement, xmlWriterWrite, xmlWriterWriteElement
1.6. XML API

1.6.108 xmlWriterEndPI

SYNOPSIS

```
int xmlWriterEndPI(xmlWriter writer)
```

Parameters:

- `writer` - the xmlWriter

Returns:

- the bytes written (may be 0 because of buffering) or -1 in case of error

DESCRIPTION

`xmlWriterEndPI()` ends the current processing instruction being written.

EXAMPLE

```
$ret = (xmlWriterEndPI($writer))
```

SEE ALSO

`xmlWriterStartPI`, `xmlWriterWrite`, `xmlWriterWritePI`
### 1.6.109 xmlWriterGetContent

**SYNOPSIS**

```
string xmlWriterGetContent(xmlWriter writer)
```

**Parameters:**

- **writer** - the xmlWriter

**Returns:**

- the data that has been written to `writer`

**DESCRIPTION**

If `writer` was created using `xmlWriterNewToString()`, then `xmlWriterGetContent()` returns the output of `writer`. No output will be returned for writers created with `xmlWriterNewToFile()`.

**EXAMPLE**

```
<output = (xmlWriterGetContent($writer))>
```

**SEE ALSO**

`xmlWriterEndDocument`
1.6.110  xmlWriterNewToFile

SYNOPSIS

    xmlWriterNewToFile(string filename)

Parameters:

    • filename - the file to write the XML document to

Returns:

    • the new xmlWriter object

DESCRIPTION

xmlWriterNewToFile() starts a new xmlWriter with the output going to the file filename.

EXAMPLE

    <$writer = (xmlWriterNewToFile($filename))>

SEE ALSO

xmlWriterNewToString, xmlWriterSetIndent, xmlWriterStartDocument
1.6.111 xmlWriterNewToString

SYNOPSIS

```xml
xmlWriter xmlWriterNewToString(string options)
```

Parameters:

- `options` - currently no options, use the empty string ''

Returns:

- the new xmlWriter object

DESCRIPTION

xmlWriterNewToString() starts a new xmlWriter that will output its results to a string in Vortex, as opposed to writing to them to a file. This output can be retrieved with xmlWriterGetContent().

EXAMPLE

```xml
<?$writer = (xmlWriterNewToString('' ))>
```

SEE ALSO

xmlWriterNewToFile, xmlWriterSetIndent, xmlWriterStartDocument
### 1.6.112 xmlWriterSetIndent

#### SYNOPSIS

```c
int xmlWriterSetIndent(xmlWriter writer, int indent [, string indentString])
```

**Parameters:**

- **writer** - the `xmlWriter`
- **indent**
  - 0 (*default*) - do not indent
  - 1 - indent output
- **indentString** (*optional*) - What string to use for indenting. The default is a single space.

**Returns:**

- 0 - success
- -1 - error

#### DESCRIPTION

`xmlWriterSetIndent()` sets writer to automatically apply line breaks and spaces between XML elements to give an indented structure to the XML file.

Note that all spaces between XML elements are significant and technically doing this changes the contents of the XML file (see the `xmlTree Text Nodes and Children` (p. 349) for details).

#### EXAMPLE

```c
$ret = (xmlWriterSetIndent($writer, 1, ' '))
```

#### SEE ALSO

`xmlWriterNewToFile`, `xmlWriterNewToString`, `xmlWriterStartDocument`
1.6.113  xmlWriterStartAttribute

SYNOPSIS

    int xmlWriterStartAttribute(xmlWriter writer, string name
                      [, string prefix [, string URI]])

Parameters:

    • writer - the xmlWriter
    • name - the name of the attribute
    • prefix (optional) - the XML namespace prefix for the attribute
    • URI (optional) - the URI for the namespace

Returns:

    • the bytes written (may be 0 because of buffering) or -1 in case of error

DESCRIPTION

xmlWriterStartAttribute() starts writing a new attribute to writer. If this will be a simple 'start attribute-write content-end attribute' sequence, you can do it with one call to xmlWriterWriteAttribute() instead.

EXAMPLE

    <$ret = (xmlWriterStartAttribute($writer, 'price'))>

SEE ALSO

xmlWriterWrite, xmlWriterEndAttribute, xmlWriterWriteAttribute,
1.6.114 xmlWriterStartCDATA

SYNOPSIS

    int xmlWriterStartCDATA(xmlWriter writer)

Parameters:

    • writer - the xmlWriter

Returns:

    • the bytes written (may be 0 because of buffering) or -1 in case of error

DESCRIPTION

xmlWriterStartCDATA() starts a new CDATA block in writer. If this will be a simple 'start CDATA-write content-end CDATA' sequence, you can do it with one call to xmlWriterWriteCDATA() instead.

EXAMPLE

    <$ret = (xmlWriterStartCDATA($writer))>

SEE ALSO

xmlWriterWrite, xmlWriterEndCDATA, xmlWriterWriteCDATA,
1.6.115 xmlWriterStartComment

SYNOPSIS

```c
int xmlWriterStartComment(xmlWriter writer)
```

Parameters:

- `writer` - the xmlWriter

Returns:

- the bytes written (may be 0 because of buffering) or -1 in case of error

DESCRIPTION

`xmlWriterStartComment()` starts a new comment in `writer`. If this will be a simple 'start comment-write content-end comment' sequence, you can do it with one call to `xmlWriterWriteComment()` instead.

EXAMPLE

```$ret = (xmlWriterStartComment($writer))>
```

SEE ALSO

`xmlWriterWrite, xmlWriterEndComment, xmlWriterWriteComment`
1.6. XML API

1.6.116  xmlWriterStartDocument

SYNOPSIS

```c
int xmlWriterStartDocument(xmlWriter writer, string version
[, string encoding [, string standalone]])
```

Parameters:

- `writer` - the xmlWriter
- `version` - the XML version to use (usually 1.0)
- `encoding` (optional) - what character encoding to use for the document. UTF-8 is used by default.
- `standalone` (optional) - pass `yes` to make this a standalone XML document. See http://www.w3.org/TR/REC-xml/#sec-rmd for more information. The value `no` may be passed to make the document not standalone (default).

Returns:

- the bytes written (may be 0 because of buffering) or -1 in case of error

DESCRIPTION

`xmlWriterStartDocument()` starts the XML document in `writer`. This consists of writing out the XML declaration, such as `<?xml version="1.0" encoding="UTF-8"?>`

EXAMPLE

```c
<$ret = (xmlWriterStartDocument($writer, '1.0', 'UTF-8'))>
```

SEE ALSO

`xmlWriterNewToFile`, `xmlWriterNewToString`, `xmlWriterSetIndent`
1.6.117 xmlWriterStartElement

SYNOPSIS

    int xmlWriterStartElement(xmlWriter writer, string name
        [, string prefix [, string URI]] )

Parameters:

• writer - the xmlWriter
• name - the name of the new element
• prefix (optional) - the namespace prefix to use for this element. Default is no prefix.
• URI (optional) - declare the prefix namespace to be URI

Returns:

• the bytes written (may be 0 because of buffering) or -1 in case of error

DESCRIPTION

xmlWriterStartElement() starts a new element in writer. If this will be a simple 'start element-write content-end element' pattern, you can do it with one call to xmlWriterWriteElement() instead.

EXAMPLE

    <$ret = (xmlWriterStartElement($writer, 'item'))>

SEE ALSO

xmlWriterWrite, xmlWriterEndElement, xmlWriterWriteElement
1.6.118 xmlWriterStartPI

SYNOPSIS

```c
int xmlWriterStartPI(xmlWriter writer, string target)
```

Parameters:

- **writer** - the xmlWriter
- **target** - the application to which the processing instruction is directed

Returns:

- the bytes written (may be 0 because of buffering) or -1 in case of error

DESCRIPTION

`xmlWriterStartPI()` starts a new processing instruction in `writer`. If this is going to be a simple 'start pi-write content-end pi' pattern, you can do it with one call to `xmlWriterWritePI()` instead.

EXAMPLE

```plaintext
<$ret = (xmlWriterStartPI($writer, 'xsl-stylesheet'))>
```

SEE ALSO

`xmlWriterWrite`, `xmlWriterEndPI`, `xmlWriterWritePI`
1.6.119  xmlWriterWrite

SYNOPSIS

    int xmlWriterWrite(xmlWriter writer, string content)

Parameters:

- **writer** - the xmlWriter
- **content** - the content you want to write

Returns:

- the bytes written (may be 0 because of buffering) or -1 in case of error

DESCRIPTION

xmlWriterWrite() writes a string to the writer. This can be used for writing the contents of attributes, elements, comments, etc.

Note that the content is automatically XML-escaped. If you provided the content `one & two`, then `one & two` will be written to the output. If you have a string that is already XML-escaped, you can output it with xmlWriterWriteRaw().

EXAMPLE

    <\$ret = xmlWriterWrite($writer, 'item content')>

SEE ALSO

xmlWriterWriteRaw
1.6.120  xmlWriterWriteAttribute

SYNOPSIS

    int xmlWriterWriteAttribute(xmlWriter writer, string name,  
                               string content [, string prefix [, string URI]])

Parameters:

- writer - the xmlWriter
- name - the name of the attribute
- content - the content of the attribute
- prefix (optional) - the namespace prefix of the attribute
- URI (optional) - the namespace URI for prefix

Returns:

- the bytes written (may be 0 because of buffering) or -1 in case of error

DESCRIPTION

xmlWriterWriteAttribute() writes out a whole attribute. It is the same as calling

    xmlWriterStartAttribute($writer, $name [, $prefix [, $uri]])  
    xmlWriterWrite($writer, $content)  
    xmlWriterEndAttribute($writer)

EXAMPLE

    <$ret = (xmlWriterWriteAttribute($writer, ’price’, 5))>

SEE ALSO

xmlWriterStartAttribute, xmlWriterWrite, xmlWriterEndAttribute
### 1.6.121 xmlWriterWriteCDATA

#### SYNOPSIS

```c
int xmlWriterWriteCDATA(xmlWriter writer, string content)
```

**Parameters:**
- `writer` - the xmlWriter
- `content` - the content of the CDATA section

**Returns:**
- the bytes written (may be 0 because of buffering) or -1 in case of error

#### DESCRIPTION

`xmlWriterWriteCDATA()` writes out a complete CDATA section. It is the same as calling

```c
xmlWriterStartCDATA($writer)
xmlWriterWrite($writer, $content)
xmlWriterEndCDATA($writer)
```

#### EXAMPLE

```c
$ret = (xmlWriterWriteCDATA($writer,
   'No parsing me! &&& is ok!'))
```

#### SEE ALSO

`xmlWriterStartCDATA, xmlWriterWrite, xmlWriterEndCDATA`
1.6.122 xmlWriterWriteComment

SYNOPSIS

    int xmlWriterWriteComment(xmlWriter writer, string content)

Parameters:

• writer - the xmlWriter
• content - the content of the CDATA section

Returns:

• the bytes written (may be 0 because of buffering) or -1 in case of error

xmlWriterWriteComment() writes out a complete comment. It is the same as calling

    xmlWriterStartComment($writer)
    xmlWriterWrite($writer, $content)
    xmlWriterEndComment($writer)

EXAMPLE

    <$ret = (xmlWriterWriteComment($writer, 'This is important'))>
1.6.123 xmlWriterWriteElement

SYNOPSIS

```c
int xmlWriterWriteElement(xmlWriter writer, string name,
                         string content [, string prefix [, string URI]])
```

Parameters:

- `writer` - the xmlWriter
- `name` - the name of the element
- `content` - the content of the element
- `prefix (optional)` - the namespace prefix of the element
- `URI (optional)` - the namespace URI for prefix

Returns:

- the bytes written (may be 0 because of buffering) or -1 in case of error

DESCRIPTION

`xmlWriterWriteElement()` writes out a whole element. It is the same as calling

```c
xmlWriterStartElement($writer, $name [, $prefix [, $uri]])
xmlWriterWrite($writer, $content)
xmlWriterEndElement($writer)
```

EXAMPLE

```c
<$ret = (xmlWriterWriteElement($writer, 'item',
                            'Look at me!'))>
```
1.6.124 xmlWriterWritePI

SYNOPSIS

    int xmlWriterWritePI(xmlWriter writer, string target, string content)

Parameters:

- writer - the xmlWriter
- target - the application to which the processing instruction is directed
- content - the contents of the processing instruction

Returns:

- the bytes written (may be 0 because of buffering) or -1 in case of error

DESCRIPTION

xmlWriterWritePI() writes a complete processing instruction to writer. It is the same as calling

    xmlWriterStartPI($writer, $target)
    xmlWriterWrite($content)
    xmlWriterEndPI($writer)

EXAMPLE

    <$ret = (xmlWriterWritePI($writer, 'xml-stylesheet',
                        'type="text/xsl" href="default.xsl"'))>

SEE ALSO

xmlWriterStartPI, xmlWriterWrite, xmlWriterEndPI
xmlWriterWriteRaw

SYNOPSIS

int xmlWriterWriteRaw(xmlWriter writer, string content)

Parameters:

- writer - the xmlWriter
- content - the raw content to write to writer

Returns:

- the bytes written (may be 0 because of buffering) or -1 in case of error

DESCRIPTION

xmlWriterWriteRaw() writes the string content to writer, without applying any markup. This is useful if you have some data that is already XML escaped, such as 5 &gt; 3 and you want to write it to the file exactly as that.

EXAMPLE

<$ret = (xmlWriterWriteRaw($writer, 'is 5 &gt; three? Yup!'))>

CAVEATS

xmlWriterWriteRaw() will write whatever you give it, including invalid XML markup. If you call something like

<$ret = (xmlWriterWriteRaw($writer, '&lt;&wheeeee!&lt;&amp;&lt;&gt;'))>

you will be left with an XML file that is not well-formed (assuming you weren’t already in a CDATA section, of course).

xmlWriterWrite would properly escape the content for storage.

SEE ALSO

xmlWriterWrite
1.6.126  xsltApplyStylesheet

SYNOPSIS

```
xmlDoc xsltApplyStylesheet(xmlDoc doc, xsltStylesheet style
[, string[] names, string[] values] )
```

Parameters:

- **doc** - the XML document to you want to transform
- **style** - the XSL stylesheet that will be used to transform the doc
- **names** *(optional)* - an array of parameter names to pass in to the transformation
- **values** *(optional)* - an array of parameter values that correspond to the names

Returns:

- The output of the transformation as an xmlDoc

DESCRIPTION

xsltApplyStylesheet applies the stylesheet `style` to the XML document `doc`. The result is returned as an xmlDoc object, which can be printed or saved like any other XML document. Any number of name/value pairs of parameters may be passed in to the stylesheet, which it uses as needed. If `names` and `values` are arrays, they must have the same number items.

See the `xmlTree10_XSL` sample script for examples of using the XSL engine and using parameters.

EXAMPLE

```
<result = (xsltApplyStylesheet($doc, $style))>
<names = "lang">
<values = "'EN'">
<resultParam = (xsltApplyStylesheet($doc, $style,
    $name, $value))>
```

SEE ALSO

xsltParseStylesheetDoc, xsltParseStylesheetFile, xsltParseStylesheetString
1.6.127 xsltParseStylesheetDoc

SYNOPSIS

xsltStylesheet xsltParseStylesheetDoc(xmlDoc doc)

Parameters:

- doc - the XML document to build the stylesheet from

Returns:

- the new xsltStylesheet

DESCRIPTION

xsltParseStylesheetDoc is used to create a new xsltStylesheet from an XML document that has already been parsed as an xmlDoc, or that was created dynamically. xsltParseStylesheetFile and xsltParseStylesheetString are usually used instead.

EXAMPLE

<$style = (xsltParseStylesheetDoc($doc))>

SEE ALSO

xsltApplyStylesheet, xsltParseStylesheetFile, xsltParseStylesheetString
1.6. XML API

1.6.128  xsltParseStylesheetFile

SYNOPSIS

\[
xsltStylesheet xsltParseStylesheetFile(string filename)
\]

Parameters:

- **doc** - the XSL file to read

Returns:

- the new xsltStylesheet

DESCRIPTION

xsltParseStylesheetFile is used to create a new xsltStylesheet from an XSL stylesheet on disk. The xsltStylesheet can then be used to transform XML documents with the xsltApplyStylesheet.

EXAMPLE

\[
<$style = (xsltParseStylesheetFile( 'style.xsl'))>
\]

SEE ALSO

xsltApplyStylesheet, xsltParseStylesheetDoc, xsltParseStylesheetString
1.6.129 xsltParseStylesheetString

SYNOPSIS

\[
\text{xsltStylesheet xsltParseStylesheetString(string xsl)}
\]

Parameters:

- \text{doc} - the XSL data to parse

Returns:

- the new \text{xsltStylesheet}

DESCRIPTION

\text{xsltParseStylesheetDoc} is used to create a new \text{xsltStylesheet} from XSL stylesheet data in a string. If you’re reading the string from a file, you can use \text{xsltParseStylesheetFile} instead.

EXAMPLE

\[
<\text{capture}>
<\text{xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html><body>I’m transformed!</body></html>
  </xsl:template>
</xsl:stylesheet>
</capture>

<\text{xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html><body>I’m transformed!</body></html>
  </xsl:template>
</xsl:stylesheet>

\text{SEE ALSO}

\text{xsltApplyStylesheet.xsltParseStylesheetDoc.xsltParseStylesheetFile}
1.7 MIME API

1.7.1 Overview

The Vortex MIME API, added in Texis version 7.02, provides a set of functions for parsing MIME (Multimedia Internet Mail Extensions) files, i.e. Internet mail messages. The API is implemented as a set of SQL functions which return and manipulate in-memory objects, for maximum code flexibility – the API is used almost exclusively in Vortex, not SQL. There are also some standalone functions for parsing headers, aliases and other strings, outside of a message.

1.7.2 Data Types

The MIME API has several data types. While these are returned from SQL functions, like the XML API data types (p. 336) they are memory-only objects that can only be used in Vortex, not stored externally in SQL tables.

In discussing the MIME API data types, the following MIME terms are used:

message: A complete or “top-level” Internet message, possibly containing multiple parts.

entity: One of the parts in a multipart message, or a complete message itself. An entity may itself contain further entities, which may be referred to as an entity tree.

Due to the recursive nature of MIME, these definitions are necessarily circular; see RFCs 822, 2045 and 5322 for more details. Since a “plain”, non-MIME, RFC 822 message is also MIME-compliant, this API can also parse such messages; it is not restricted to explicit “MIME-Version”-labelled messages.

Like the XML API (p. 336), the MIME API data types need not be explicitly “closed” when done – in fact there are no “close” functions. An object is freed when the last Vortex variable reference to it is freed, either by explicit reassignment to something else (or nothing), or by the variable going out of scope. (Thus, MIME API objects used only within a single function call are generally assigned to local Vortex variables, so they will be automatically freed when the call ends.) Similarly, copying an object from one Vortex variable to another does not duplicate the object, but merely increases its reference count: both variables may continue to manipulate the same underlying object, and each variable’s changes are reflected in the other.

mimeReader

The mimeReader data type parses messages, and is the first type encountered when using the API. It is created by opening a file or string containing a message, or by “opening” a mimeEntity object (since MIME messages can recurse). A mimeReader object parses the MIME message as a stream, internally maintaining a current-entity cursor, and there are functions to iterate over all the entities in a message.

Since a mimeReader object usually has a current entity it points to in the message, the mimeEntity access functions for obtaining headers, body etc. also work on mimeReader objects as a convenience, to avoid having to first get a mimeEntity object from the reader.
mimeEntity

The mimeEntity object contains a MIME entity. It can be created (obtained) from a mimeReader object at its current cursor position. There are access functions for obtaining information about the entity, such as headers, body, etc.

Since these access functions also work on mimeReader objects, typically they are used directly on the mimeReader for convenience, without ever explicitly creating a mimeEntity. However, if an entity or entity tree of a message must be “copied off” for separate handling elsewhere while the rest of the message continues to be parsed, a mimeEntity can be created at that point to contain such an entity. A new mimeReader could then be created to parse that separated entity.

Example Script

The following is a small example script that illustrates use of the MIME API. It parses a MIME message file given by the variable file on the command line, printing a one line summary of each entity. See the following pages for details on the functions used here.

<script language=vortex>

<a name=printEntity entity>
<!-- Prints a summary of given $entity. -->
<local depth imap contentType filename text>

<$depth = (mimeEntityGetDepth($entity))>
<$imap = (mimeEntityGetImapSectionSpecification($entity))>
</a>

<a name=main public file>
<local reader>

<$reader = (mimeReaderOpenFile($file))>
</a>
</script>
Given the following example MIME message:

From: "John Smith" <john.smith@acme.com>
To: <mary.jones@acme.com>
Subject: MIME Test
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="boundayMixed"

Preamble of multipart/mixed message.

--boundayMixed
Content-Type: multipart/related; boundary="boundayRelated"

Preamble of multipart/related entity.

--boundayRelated
Content-Type: multipart/alternative; boundary="boundayAlternative"

Preamble of multipart/alternative entity.

--boundayAlternative
Content-Type: text/plain; charset="us-ascii"

Plain text version of main message.

--boundayAlternative
Content-Type: text/html; charset="us-ascii"
Content-Transfer-Encoding: quoted-printable

<body>HTML version of main message, with an inline image.
<img src=3D"cid:image001.gif@01CB5A58.A51AF3E0"></body>

--boundayAlternative--

--boundayRelated
Content-Type: image/gif; name="image001.gif"
Content-Transfer-Encoding: base64
Content-ID: <image001.gif@01CB5A58.A51AF3E0>

R0lGODlhAQABAIDA==

--boundayRelated--

--boundayMixed
Content-Type: image/gif; name="attachment.gif"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="attachment.gif"

R0lGODlhAQABAIDA==

--boundayMixed--
Running the example script above on the example MIME message above will produce the following output:

0 multipart/mixed [part.bin] Preamble of multipart/mixed message...
  1.0 multipart/related [part-2.bin] Preamble of multipart/related...
    1.1.0 multipart/alternative [part-3.bin] Preamble of multipart/alternative...
      1.1.1 text/plain [part.txt] Plain text version of main message...
      1.1.2 text/html [index.html] HTML version of main message, ...
  1.2 image/gif [image001.gif] GIF89a....
  2 image/gif [attachment.gif] GIF89a....
SYNOPSIS

varbyte mimeEntityGetBody(mimeEntity entityOrReader[, varchar flags])

Parameters:

- entityOrReader - A mimeEntity or mimeReader object
- flags - An optional CSV list of flags:
  - "reparented" - Re-parent HTML if appropriate

Returns:

- The decoded varbyte body of the entity

DESCRIPTION

The mimeEntityGetBody function returns the varbyte body of the given entity, with any transfer-encoding decoded. A varbyte value is returned instead of varchar in case the body is binary, e.g. an image.

If the optional “reparented” flag is given, then if the body is HTML and within a multipart/relatedentity tree, links within the HTML will be reparented with relatedfiles mode (p. 235): all internal links (to other parts of the message) will be changed to their safe filenames (see mimeEntityGetSafeFilename, p. 533), and all external links will be made absolute. Thus, if all the parts of the message have their reparented mimeEntityGetBody values written to their safe filenames in the same directory, inline images will be resolved when the reparented HTML is viewed in a web browser. Both cid: (content ID) and ordinary (content location e.g. http:) internal links are handled. Relative links whose base (absolute) URL cannot be established (i.e. missing Content-Location) will be made relative to the fictional URL thismessage, per RFC 2557. This prevents their inadvertent access in the wrong context (e.g. relative to a display site).

Note: to make the reparented body “safe” – i.e. suppress any references to external images when viewed in a browser – the HTML can be further processed with the hideexternal mode (p. 235) of <urlcp reparentmode>, with a local re-fetch of the body HTML.

The “reparented” flag has no effect if the body is not HTML, or if the entity is not within a multipart/relatedentity tree.

EXAMPLE
<$ret = (mimeEntityGetBody($reader))>
Body: $ret

CAVEATS

SEE ALSO

mimeEntityGetSafeFilename, mimeEntityGetText
1.7. MIME API

1.7.4 mimeEntityGetChildNumber

SYNOPSIS

int mimeEntityGetChildNumber(mimeEntity entityOrReader)

Parameters:

• entityOrReader - A mimeEntity or mimeReader object

Returns:

• int child number of the entity

DESCRIPTION

The mimeEntityGetChildNumber function returns the sequence number (in source message order) of the current entity relative to its siblings, starting with 0. I.e. the first “payload” child entity (after the preamble) of a multipart is child number 0, the second is child 1, etc. If a child entity is another multipart entity, its descendant entities will in turn will be numbered starting over at 0. The very first entity (preamble) of the source message is considered the 0th child of a non-existent parent, for consistent numbering.

EXAMPLE

<\$ret = (mimeEntityGetChildNumber($reader))>

For the example MIME message above (p. 511), each multipart/... preamble would be child 0, as each is the first child of its parent. The text/html entity would be child 1, as it is the second entity of its parent (the multipart/alternative entity).

CAVEATS

SEE ALSO

mimeEntityGetImapSectionSpecification, mimeEntityIsFirstChild, mimeEntityGetSequenceNumber
1.7.5 mimeEntityGetContentLocation

SYNOPSIS

varchar mimeEntityGetContentLocation(mimeEntity entityOrReader)

Parameters:

- entityOrReader - A mimeEntity or mimeReader object

Returns:

- The varchar content location URL of the entity

DESCRIPTION

The mimeEntityGetContentLocation function returns the absolute URL for the content location of the entity. This is derived from its Content-Location header, taking into account ancestor entities’ headers if needed. If the content location cannot be determined, a URL of “thismessage:/*” may be returned.

EXAMPLE

<$ret = (mimeEntityGetContentLocation($reader))>
1.7.6 mimeEntityGetDepth

SYNOPSIS

int mimeEntityGetDepth(mimeEntity entityOrReader)

Parameters:

  • entityOrReader - A mimeEntity or mimeReader object

Returns:

  • int depth of the entity from root

DESCRIPTION

The mimeEntityGetDepth function returns the integer depth of the entity, in levels from the root entity of the message. Thus a plain RFC 822 message (single entity) is depth 0, as is the top-level preamble of a multipart message.

EXAMPLE

<$ret = (mimeEntityGetDepth($reader))>

CAVEATS

SEE ALSO

mimeEntityGetChildNumber
1.7.7 mimeEntityGetHeaderNames

SYNOPSIS

varstrlst mimeEntityGetHeaderNames(mimeEntity entityOrReader)

Parameters:

• entityOrReader - A mimeEntity or mimeReader object

Returns:

• varstrlst list of header names

DESCRIPTION

The mimeEntityGetHeaderNames function returns the unique names of headers in the entity, as a varstrlst list.

EXAMPLE

$$ret = (mimeEntityGetHeaderNames($reader))$$
Headers are named: <loop $ret> $ret </loop>
<fmt "\n">

CAVEATS

The multi-value varstrlst return value will be split into multiple Vortex variable values that can be looped over, but only if <sqlcp arrayconvert> (p. 127) is on (the default).

SEE ALSO

mimeEntityGetHeaderValue
1.7.8 mimeEntityGetHeaderParameterNames

SYNOPSIS

```python
varstrlst mimeEntityGetHeaderParameterNames(mimeEntity entityOrReader,
                                          varchar headerName)
```

Parameters:

- `entityOrReader` - A mimeEntity or mimeReader object
- `headerName` - The varchar name of the header

Returns:

- `varstrlst` list of parameter names for header

DESCRIPTION

The `mimeEntityGetHeaderParameterNames` function returns the unique names of parameters of all header(s) named `headerName` in the entity, as a `varstrlst` list.

EXAMPLE

```plaintext
<$ret = (mimeEntityGetHeaderParameterNames($reader, "Content-Type" ))>
Content-Type params are named: <loop $ret> $ret </loop><fmt "\n">
```

CAVEATS

The multi-value `varstrlst` return value will be split into multiple Vortex variable values that can be looped over, but only if `<sqlcp arrayconvert>` (p. 127) is on (the default).

SEE ALSO

mimeEntityGetHeaderParameterValues
1.7.9 mimeEntityGetHeaderParameterValues

SYNOPSIS

varstrlst mimeEntityGetHeaderParameterValues(mimeEntity entityOrReader,
   varchar headerName [, varchar paramName])

Parameters:

- entityOrReader - A mimeEntity or mimeReader object
- headerName - The varchar name of the header
- paramName - The optional varchar name of the parameter

Returns:

- varstrlst list of parameter values from header for given parameter name

DESCRIPTION

The mimeEntityGetHeaderParameterValues function returns the values of parameters named paramName from the header named headerName in the entity, RFC 2047 encoded-word decoded, as a varstrlst list. If paramName is not given or is empty, the first/main value of the header is returned (i.e. the “empty/unnamed” parameter; see example).

EXAMPLE

<$ret = (mimeEntityGetHeaderParameterValues($reader, "Content-Type" ))>
Content-Type is: $ret
<$ret = (mimeEntityGetHeaderParameterValues($reader, "Content-Type", "charset"))>
Charset is: $ret

Given the entity header "Content-Type: text/html; charset="ISO-8859-1"", the output would be:

Content-Type is: text/html
Charset is: ISO-8859-1
CAVEATS

The multi-value `varstrlst` return value will be split into multiple Vortex variable values that can be looped over, but only if `<sqlcp arrayconvert>` (p. 127) is on (the default).

SEE ALSO

mimeEntityGetHeaderParameterNames
1.7.10 mimeEntityGetHeaderValues

SYNOPSIS

varstrlst mimeEntityGetHeaderValues(mimeEntity entityOrReader,
                      varchar headerName)

Parameters:

- entityOrReader - A mimeEntity or mimeReader object
- headerName - The varchar name of the header

Returns:

- varstrlst list of decoded value(s) of given header name

DESCRIPTION

The mimeEntityGetHeaderValues function returns the full values (i.e. including parameters) of
header(s) named headerName in the entity, unfolded and RFC 2047 encoded-word decoded, as a
varstrlst list.

Note: To obtain just the “empty/unnamed parameter” value – e.g. the MIME type without parameters, from
a Content-Type header – use mimeEntityGetHeaderParameterValues (p. 520).

EXAMPLE

<$ret = (mimeEntityGetHeaderValues($reader, "To" ))>
Entity is addressed to: $ret

CAVEATS

The multi-value varstrlst return value will be split into multiple Vortex variable values that can be
looped over, but only if <sqlcp arrayconvert> (p. 127) is on (the default).

SEE ALSO

mimeEntityGetHeaderNames, mimeEntityGetHeaderParameterValues
1.7.11  mimeEntityGetImapSectionSpecification

SYNOPSIS

varchar mimeEntityGetImapSectionSpecification(mimeEntity entityOrReader)

Parameters:

- entityOrReader - A mimeEntity or mimeReader object

Returns:

- varchar IMAP section specification

DESCRIPTION

The mimeEntityGetImapSectionSpecification function returns the IMAP section specification (number) for the given entity. This is a dotted-decimal number used by IMAP to denote entities (sections) within a message.

EXAMPLE

```plaintext
$ret = (mimeEntityGetImapSectionSpecification($reader))
IMAP section number: $ret
```

See the MIME example script (p. 510) for a more thorough example.

SEE ALSO

mimeEntityGetChildNumber
1.7.12 mimeEntityGetMessageFilename

SYNOPSIS

`varchar mimeEntityGetMessageFilename(mimeEntity entityOrReader)`

Parameters:

- `entityOrReader` - A `mimeEntity` or `mimeReader` object

Returns:

- `varchar` filename for the entity, as specified in the message

DESCRIPTION

The `mimeEntityGetMessageFilename` function returns the filename for the entity, as specified in the message, or empty string if none. Note that this filename may contain a full or partial path, may use unsafe characters, may be missing (empty), and/or may not be unique within the message. Thus the `mimeEntityGetSafeFilename` function (p. 533) is recommended instead.

The message filename is the first value found from the following sources:

1. Path part of `Content-Location` URL
2. `Content-Type name` parameter
3. `Content-Disposition filename` parameter

EXAMPLE

```
$ret = (mimeEntityGetMessageFilename($reader))
Original filename: $ret
```

CAVEATS

The returned filename may not be safe for local filesystem use. Use `mimeEntityGetSafeFilename` (p. 533) instead.
SEE ALSO

mimeEntityGetSafeFilename
1.7.13  mimeEntityGetRawBody

SYNOPSIS

`varchar mimeEntityGetRawBody(mimeEntity entityOrReader)`

Parameters:

- `entityOrReader` - A `mimeEntity` or `mimeReader` object

Returns:

- `varchar` raw (original encoded) body of entity

DESCRIPTION

The `mimeEntityGetRawBody` function returns the `varchar` raw – i.e. original encoded – body of the entity. This can be used for debugging purposes, e.g. analyzing the structure of a message.

EXAMPLE

```<$ret = (mimeEntityGetRawBody($reader))>```

CAVEATS

Any transfer encodings applied to the body will not be removed; typically `mimeEntityGetBody` (p. 513) should be used instead, as it decodes the body.

SEE ALSO

`mimeEntityGetBody`
### 1.7.14 mimeEntityGetRawBodyOffset

**SYNOPSIS**

```
int64 mimeEntityGetRawBodyOffset(mimeEntity entityOrReader)
```

**Parameters:**

- `entityOrReader` - A `mimeEntity` or `mimeReader` object

**Returns:**

- `int64` byte offset of raw (original encoded) body in message

**DESCRIPTION**

The `mimeEntityGetRawBodyOffset` function returns the byte offset of the entity’s raw (original encoded) body in the overall message (not entity), as an `int64` value. This can be used for debugging purposes, e.g. analyzing the structure of a message.

**EXAMPLE**

```
$ret = (mimeEntityGetRawBodyOffset($reader))
```

**CAVEATS**

Typically `mimeEntityGetBody` (p. 513) is used instead, as it decodes the body.

**SEE ALSO**

`mimeEntityGetBody`
1.7.15  mimeEntityGetRawBodySize

SYNOPSIS

int64 mimeEntityGetRawBodySize(mimeEntity entityOrReader)

Parameters:

- entityOrReader - A mimeEntity or mimeReader object

Returns:

- int64 byte size of raw (original encoded) body in message

DESCRIPTION

The mimeEntityGetRawBodySize function returns the byte size of the entity’s raw (original encoded) body in the overall message (not entity), as an int64 value. This can be used for debugging purposes, e.g. analyzing the structure of a message.

EXAMPLE

<$ret = (mimeEntityGetRawBodySize($reader))>

CAVEATS

Typically mimeEntityGetBody (p. 513) is used instead, as it decodes the body.

SEE ALSO

mimeEntityGetBody
1.7.16 mimeEntityGetRawHeaderSection

SYNOPSIS

\texttt{varchar mimeEntityGetRawHeaderSection(mimeEntity entityOrReader)}

Parameters:

- \texttt{entityOrReader} - A \texttt{mimeEntity} or \texttt{mimeReader} object

Returns:

- \texttt{varchar} raw (original encoded) header section of entity

DESCRIPTION

The \texttt{mimeEntityGetRawHeaderSection} function returns the \texttt{varchar} raw (original encoded) header section (i.e. all headers) of the entity. This can be used for debugging purposes, e.g. analyzing the structure of a message.

EXAMPLE

\texttt{<$ret = (mimeEntityGetRawHeaderSection($reader))>}

CAVEATS

Typically \texttt{mimeEntityGetHeaderValues} (p. 522) is used instead, as it decodes the headers and returns only the one specified.

SEE ALSO

\texttt{mimeEntityGetHeaderValues}
1.7.17 mimeEntityGetRawHeaderSectionOffset

SYNOPSIS

int64 mimeEntityGetRawHeaderSectionOffset(mimeEntity entityOrReader)

Parameters:

• entityOrReader - A mimeEntity or mimeReader object

Returns:

• int64 byte offset of raw (original encoded) header section of entity in message

DESCRIPTION

The mimeEntityGetRawHeaderSectionOffset function returns the int64 byte offset of the raw (original encoded) header section (i.e. all headers) of the entity, as an offset into the overall message (not entity). This can be used for debugging purposes, e.g. analyzing the structure of a message.

EXAMPLE

<$ret = (mimeEntityGetRawHeaderSectionOffset($reader))>

CAVEATS

Typically mimeEntityGetHeaderValue (p. 522) is used instead, as it decodes the headers and returns only the one specified.

SEE ALSO

mimeEntityGetHeaderValue
1.7. MIME API

1.7.18 mimeEntityGetRawHeaderSectionSize

SYNOPSIS

int64 mimeEntityGetRawHeaderSectionSize(mimeEntity entityOrReader)

Parameters:

- entityOrReader - A mimeEntity or mimeReader object

Returns:

- int64 byte size of raw (original encoded) header section of entity

DESCRIPTION

The mimeEntityGetRawHeaderSectionSize function returns the int64 byte size of the raw (original encoded) header section (i.e. all headers) of the entity. This can be used for debugging purposes, e.g. analyzing the structure of a message.

EXAMPLE

<$ret = (mimeEntityGetRawHeaderSectionSize($reader))>

CAVEATS

Typically mimeEntityGetHeaderValues (p. 522) is used instead, as it decodes the headers and returns only the one specified.

SEE ALSO

mimeEntityGetHeaderValues
1.7.19  mimeEntityGetRawHeaderValues

SYNOPSIS

\texttt{varstrlst \texttt{mimeEntityGetRawHeaderValues}(mimeEntity \texttt{entityOrReader},
\hspace{1em}varchar \texttt{headerName})}

Parameters:

- \texttt{entityOrReader} - A mimeEntity or mimeReader object
- \texttt{headerName} - The varchar name of the header

Returns:

- varchar raw (original encoded) header named \textquote{headerName}’

DESCRIPTION

The \texttt{mimeEntityGetRawHeaderValues} function returns the full value(s) of the header named \texttt{headerName} from the entity, as a \texttt{varstrlst} list. Note that the value(s) returned are raw, i.e. neither unfolded nor decoded. This function can be used for debugging purposes, e.g. analyzing the structure of a message.

EXAMPLE

\texttt{$\langle$ret = (mimeEntityGetRawHeaderValues($reader, "To")$\rangle$}

CAVEATS

Typically \texttt{mimeEntityGetHeaderValues} (p. 522) is used instead, as it decodes the headers.

SEE ALSO

\texttt{mimeEntityGetHeaderValues}
1.7.20  mimeEntityGetSafeFilename

SYNOPSIS

varchar mimeEntityGetSafeFilename(mimeEntity entityOrReader)

Parameters:
- entityOrReader - A mimeEntity or mimeReader object

Returns:
- varchar filename for entity, safe for filesystem

DESCRIPTION

The mimeEntityGetSafeFilename function returns the filename for the entity. The filename is based on the original message filename (see mimeEntityGetMessageFilename, p. 524), but never contains a directory component, is always unique within the overall message, is safe for the filesystem (e.g. no device names nor control characters), and is never empty (one will be created). It will be index.ext for the start body part (e.g. of multipart/related messages).

Additionally, the returned filename will be referenced by inline images/objects in the returned value of mimeEntityGetBody (p. 513), when the “reparented” flag is given. Thus, if the reparented bodies of all the entities in a message are written out to their mimeEntityGetSafeFilename filenames in the same directory, a web browser viewing the directory should correctly display the message and any inline images. See the example below.

EXAMPLE

```<dir = "/var/www/html/theMessageDir">  
<while (mimeReaderMoveToNextEntity($reader) = '1')>
  <local path body>
    <$ret = (mimeEntityGetSafeFilename($reader))>
    <strfmt "%s/%s" $dir $ret><$path = $ret>
    <$body = (mimeEntityGetBody($reader, "reparented" ))>
    <write $path><fmt "%s" $body></write>
  </local path body>
</while>
```

SEE ALSO

mimeEntityGetMessageFilename
1.7.21  mimeEntityGetSequenceNumber

SYNOPSIS

int mimeEntityGetSequenceNumber(mimeEntity entityOrReader)

Parameters:

- entityOrReader - A mimeEntity or mimeReader object

Returns:

- int sequence number for the entity

DESCRIPTION

The mimeEntityGetSequenceNumber function returns the sequence number for the entity. This is an ordinal integer number starting with zero and incrementing by one for each entity; it identifies the order of the entity within the overall message.

EXAMPLE

<$ret = (mimeEntityGetSequenceNumber($reader))>

CAVEATS

Another mimeReader object opened on the same message – but with a different max depth – may number the entities differently, as entities at greater than max depth will be grouped into one entity tree, but may be separated in the other reader (if greater max depth).

SEE ALSO

mimeEntityGetChildNumber
1.7.22 mimeEntityGetText

SYNOPSIS

`varchar mimeEntityGetText(mimeEntity entityOrReader)`

Parameters:

- `entityOrReader` - A mimeEntity or mimeReader object

Returns:

- `varchar` formatted text of the entity’s body

DESCRIPTION

The `mimeEntityGetText` function returns the formatted text of the entity’s body, i.e. the equivalent of calling `<urlinfo text>` (p. 193) for a fetched page. E.g. HTML tags and entities will be removed and translated if the body MIME type is `text/html`. The text is typically returned in UTF-8 if the entity body source charset was recognized and supported; see `mimeEntityGetTextCharset` (p. 536).

EXAMPLE

```php
<$ret = (mimeEntityGetText($reader))>
Text of entity: $ret
```

SEE ALSO

mimeEntityGetBody
1.7.23  mimeEntityGetTextCharset

SYNOPSIS

varchar mimeEntityGetTextCharset(mimeEntity entityOrReader)

Parameters:

• entityOrReader - A mimeEntity or mimeReader object

Returns:

• varchar charset of mimeEntityGetText return data

DESCRIPTION

The mimeEntityGetTextCharset function returns the charset of the data returned by mimeEntityGetText, or empty string if unknown. If the entity body’s source charset was recognized and supported during formatting, the text charset will generally be UTF-8.

EXAMPLE

<$ret = (mimeEntityGetTextCharset($reader))>
Charset of text of entity: $ret
<$ret = (mimeEntityGetText($reader))>
Text of entity: $ret

SEE ALSO

mimeEntityGetText
1.7.24 mimeEntityGetTextFormatter

SYNOPSIS

varchar mimeEntityGetTextFormatter(mimeEntity entityOrReader)

Parameters:

• entityOrReader - A mimeEntity or mimeReader object

Returns:

• varchar token indicating text formatter used for body text

DESCRIPTION

The mimeEntityGetTextFormatter function returns a string token indicating which internal formatter was used to format the entity's body for mimeEntityGetText (p. 535). These are the same tokens that <urlinfo textformatter> returns; see p. 193 for a list.

EXAMPLE

    <$ret = (mimeEntityGetTextFormatter($reader))>

SEE ALSO

mimeEntityGetText
1.7.25  mimeEntityIsLastChild

SYNOPSIS

int mimeEntityIsLastChild(mimeEntity entityOrReader)

Parameters:

•  entityOrReader - A mimeEntity or mimeReader object

Returns:

•  int boolean value indicating whether entity is the last child of its parent

DESCRIPTION

The mimeEntityIsLastChild function returns an integer boolean value (1 for true, 0 for false) indicating whether the entity is the last child of its parent (immediate ancestor). This can be used in a mimeReaderMoveToNextEntity loop to perhaps perform actions needed after all children at a given level are parsed.

EXAMPLE

<$ret = (mimeEntityIsLastChild($reader))>

SEE ALSO

mimeEntityGetChildNumber
### 1.7.26 mimeEntityIsReparented

#### SYNOPSIS

```c
int mimeEntityIsReparented(mimeEntity entityOrReader)
```

**Parameters:**
- `entityOrReader` - A `mimeEntity` or `mimeReader` object

**Returns:**
- An integer boolean value indicating whether the value returned by `mimeEntityGetBody` with the "reparented" flag, is actually reparented, i.e. is HTML and within a `multipart/related` entity tree.

#### DESCRIPTION

The `mimeEntityIsReparented` function returns an integer boolean value (1 if true, 0 if false) that indicates whether the value returned by `mimeEntityGetBody` (p. 513) with the "reparented" flag, is actually reparented, i.e. is HTML and within a `multipart/related` entity tree.

#### EXAMPLE

```c
<$ret = (mimeEntityIsReparented($reader))>
```

#### SEE ALSO

`mimeEntityGetBody`
1.7.27  mimeEntityIsStartBodyPart

SYNOPSIS

int mimeEntityIsStartBodyPart(mimeEntity entityOrReader)

Parameters:

- entityOrReader - A mimeEntity or mimeReader object

Returns:

- int boolean value indicating whether the entity is the overall message start part

DESCRIPTION

The mimeEntityGet function returns an integer boolean value (1 if true, 0 if false) indicating whether the entity is the start body part for the overall message. I.e. in a multipart/related message, one entity is the actual message – the start body part – and the others are inline images or other associated objects. The start body part is always named index.ext by the mimeEntityGetSafeFilename function (p. 533), so that if all the message’s entities are written to the same directory and the directory is accessed via the web, the start body part will be returned by the web server.

An appropriate start body part is always chosen for every overall message, even if the overall type is not multipart/related.

EXAMPLE

<$ret = (mimeEntityIsStartBodyPart($reader))>

SEE ALSO

mimeEntityGetSafeFilename
1.7.28 mimeReaderGetEntity

SYNOPSIS

mimeEntity mimeReaderGetEntity(mimeReader reader)

Parameters:

• reader - A mimeReader object

Returns:

• mimeEntity object for current entity

DESCRIPTION

The mimeReaderGetEntity function returns the mimeEntity object at the reader's cursor (current entity location). This object can be saved for future reference after the reader has moved on to other entities.

EXAMPLE

<$ret = (mimeReaderGetEntity($reader))>

SEE ALSO

mimeReaderGetFullEntity
1.7.29 mimeReaderGetFullEntity

SYNOPSIS

mimeEntity mimeReaderGetFullEntity(mimeReader reader)

Parameters:

- reader - A mimeReader object

Returns:

- mimeEntity object for entire entity tree at cursor

DESCRIPTION

The `mimeReaderGetFullEntity` function returns the `mimeEntity` object for the full entity tree at the reader's cursor. This object can be saved for future reference after the reader has moved on to other entities. Unlike the value returned by `mimeReaderGetEntity` (p. 541), this entity holds the entire entity tree at the reader's location – regardless of max depth for the reader – not just the “node” atomic entity. Thus it can be used to open a new reader to parse just that sub-tree of the message.

EXAMPLE

```php
<$entity = (mimeReaderGetFullEntity($reader))>
<$newReader = (mimeReaderOpenEntity($entity, -1, "inherittreeposition"))>
```

SEE ALSO

`mimeReaderGetEntity`, `mimeReaderOpenEntity`
1.7.30 mimeReaderMoveToNextEntity

SYNOPSIS

int mimeReaderMoveToNextEntity(mimeReader reader)

Parameters:

- reader - A mimeReader object

Returns:

- int value: 1 on success, 0 on EOF, -1 on error

DESCRIPTION

The `mimeReaderMoveToNextEntity` function moves the reader cursor (current entity location) to the next entity in the message. It returns 1 if successful (new entity reached), 0 if end of message (no more entities), or -1 on error. The entities will be traversed in message order, i.e. pre-order.

EXAMPLE

```php
<$reader = (mimeReaderOpenFile("/tmp/someMessage"))>
<while (mimeReaderMoveToNextEntity($reader) = '1')>
  ... print current entity ...
</while>
```

SEE ALSO

mimeReaderMoveToNextEntitySibling
1.7.31 mimeReaderMoveToNextEntitySibling

SYNOPSIS

```c
int mimeReaderMoveToNextEntitySibling(mimeReader reader)
```

Parameters:

- `reader` - A mimeReader object

Returns:

- `int value`: 1 on success, 0 on EOF, -1 on error

DESCRIPTION

The `mimeReaderMoveToNextEntitySibling` function is like `mimeReaderMoveToNextEntity` (p. 543), but moves to the next “sibling” in the message tree, i.e. it skips over the current entity’s descendants (if any). It returns 1 if successful (new entity reached), 0 if end of message (no more entities), or -1 on error. The entities will be traversed in message order, i.e. pre-order.

EXAMPLE

Changing the `<main>` function of the example script above (p. 510) to the following:

```c
<local reader>
  <$reader = (mimeReaderOpenFile($file))>
  <$ret = (mimeReaderMoveToNextEntity($reader))>
  <printEntity entity=$reader> <!-- multipart/mixed preamble -->
  <$ret = (mimeReaderMoveToNextEntity($reader))>
  <printEntity entity=$reader> <!-- multipart/related preamble -->
  <while (mimeReaderMoveToNextEntitySibling($reader) = ' 1')>
    <printEntity entity=$reader>
  </while>
</local reader>
```

would yield the following output when run on the example MIME message (p. 511):

```
0 multipart/mixed [part.bin] Preamble of multipart/mixed me...
  1.0 multipart/related [part-2.bin] Preamble of multipart/related ...
  2 image/gif [attachment.gif] GIF89a....
```
SEE ALSO

mimeReaderMoveToNextEntity
1.7.32  mimeReaderOpenEntity

SYNOPSIS

mimeReader mimeReaderOpenEntity(mimeEntity entityOrReader
    [, int64 maxDepth[, varchar flags]])

Parameters:

- entityOrReader - A mimeEntity or mimeReader object
- maxDepth - An optional integer maximum depth to parse to
- flags - An optional CSV list of flags:
  – “inherittreeposition” - Continue with same depth, IMAP number, etc. of parent
  entityOrReader

Returns:

- mimeReader object

DESCRIPTION

The mimeReaderOpenEntity function returns a new mimeReader object for parsing the given entity
entityOrReader. This function can be used when an entity tree within a message must be “broken off”
and parsed by separate code that is to handle only that entity tree, independently.

The optional maxDepth parameter indicates how deep into the message tree to parse; the default of -1
indicates no limit, i.e. the tree is parsed to its leaf node entities. Entities at a depth greater than maxDepth
would be embedded as part of an entity tree returned at the max depth.

EXAMPLE

$$entity = (mimeReaderGetFullEntity($reader))$$
$$newReader = (mimeReaderOpenEntity($entity, -1,
    "inherittreeposition")$$

SEE ALSO

mimeReaderOpenFile
1.7. MIME API

1.7.33 mimeReaderOpenFile

SYNOPSIS

mimeReader mimeReaderOpenFile(varchar file[, int64 maxDepth])

Parameters:

- file - A varchar path to the MIME file to open
- maxDepth An optional integer maximum depth to parse to

Returns:

- mimeReader object for parsing the file

DESCRIPTION

The mimeReaderOpenFile function returns a new mimeReader object for parsing the MIME message in file.

The optional maxDepth parameter indicates how deep into the message tree to parse; the default of -1 indicates no limit, i.e. the tree is parsed to its leaf node entities. Entities at a depth greater than maxDepth would be embedded as part of an entity tree returned at the max depth.

EXAMPLE

```<$file = "/tmp/mimeMessage.eml"><$reader = (mimeReaderOpenFile($file))> <if "" eq $reader> Cannot open $file </if>```

CAVEATS

SEE ALSO

mimeReaderOpenString
1.7.34 mimeReaderOpenString

SYNOPSIS

mimeReader mimeReaderOpenString(varchar string[, int64 maxDepth])

Parameters:

- string - A varchar string with the MIME message
- maxDepth - An optional integer maximum depth to parse to

Returns:

- mimeReader object for parsing the message

DESCRIPTION

The mimeReaderOpenString function returns a new mimeReader object for parsing the MIME message in string.

The optional maxDepth parameter indicates how deep into the message tree to parse; the default of -1 indicates no limit, i.e. the tree is parsed to its leaf node entities. Entities at a depth greater than maxDepth would be embedded as part of an entity tree returned at the max depth.

EXAMPLE

```xml
<read "/tmp/mimeMessage.eml">
<$reader = (mimeReaderOpenString($ret))>
<if "" eq $reader>
   Cannot open MIME string
</if>
```

CAVEATS

SEE ALSO

mimeReaderOpenFile
1.7.35 headerDecode

SYNOPSIS

varchar headerDecode(varchar header)

Parameters:

- header - A raw varchar header value

Returns:

- The unfolded and decoded varchar header value

DESCRIPTION

The headerDecode function returns the given header value, unfolded (newlines removed) and with RFC 2047 encoded-words decoded. Note that regardless of charset in any encoded-words, the returned charset is always UTF-8.

EXAMPLE

<$ret = (headerDecode($subjectHdr))>

CAVEATS

The headerDecode function is for parsing standalone headers. When parsing a mail message, generally the access function mimeEntityGetHeaderValues (p. 522) is used instead.

SEE ALSO

headerGetItems
1.7.36 **headerGetItems**

**SYNOPSIS**

`varchar headerGetItems(varchar header)`

**Parameters:**

- `header` - A raw varchar header value

**Returns:**

- The raw (original encoded) list of CSV items, as a `varstrlst`

**DESCRIPTION**

The `headerGetItems` function splits the raw (original encoded) CSV header value into a list of raw (original encoded) values, returning a `varstrlst` list. Commas inside double-quoted, single-quoted, or angle-bracketed values are not considered items separators. Leading and trailing whitespace from each item is stripped.

**EXAMPLE**

```
<$toHdr = '"Smith, John" <jsmith@foo.com>, <jdoe@bar.com> Jane Doe'>
<$ret = (headerGetItems($toHdr))>
Mailboxes: <loop $ret> [$ret] </loop><fmt "\n">
```

The output would be:

```
Mailboxes: ["Smith, John" <jsmith@foo.com>] [<jdoe@bar.com> Jane Doe]
```

**CAVEATS**

The returned items are not RFC 2047 decoded, because further parsing might be done (e.g. getting parameter names and values). Use `headerDecode` (p. 549) for RFC 2047 encoded-word decoding.

**SEE ALSO**

`headerDecode`, `headerItemGetParameterValues`
### 1.7.37 headerItemGetParameterNames

**SYNOPSIS**

`varstrlst headerItemGetParameterNames(varchar headerItem)`

Parameters:
- `headerItem` - A `varchar` raw header item

Returns:
- The `varstrlst` decoded list of parameter names

**DESCRIPTION**

The `headerItemGetParameterNames` function returns a unique list of parameter names in `headerItem`, which is a single raw (original encoded) header item (i.e. one item in a CSV list, if header value is CSV, e.g. email To header). The returned names are RFC 2047 encoded-word decoded.

Note that a shortcut function exists for `mimeEntity` objects, to obtain the header and parse it for parameters in one call; see `mimeEntityGetHeaderParameterNames` (p. 519).

**EXAMPLE**

```
<$conType = 'text/html; charset="ISO-8859-1"; name="foo.html"'>
<$ret = (headerItemGetParameterNames($conType))>
Param names: <loop $ret> [$ret] </loop><fmt "\n">
```

The output would be:

Param names: [charset] [name]

**CAVEATS**

If the header being decoded is a potentially multi-item CSV list (e.g. “To” addresses), it should be split via `headerGetItems` (p. 550) first.

**SEE ALSO**

`headerItemGetParameterValues`, `mimeEntityGetHeaderParameterNames`
1.7.38 headerItemGetParameterValues

SYNOPSIS

`varstrlst headerItemGetParameterValues(varchar headerItem [, varchar paramName])`

Parameters:

- `headerItem` - A `varchar` raw header item
- `paramName` - An optional `varchar` parameter name

Returns:

- The `varstrlst` decoded list of parameter values

DESCRIPTION

The `headerItemGetParameterValues` function returns a list of parameter value(s) for the parameter named `paramName` in `headerItem`, which is a single raw (original encoded) header item (i.e. one item in a comma-separated values list, if header is CSV). The returned values are RFC 2047 encoded-word decoded.

If `paramName` is not given or is empty, the first/main value of the header is returned (i.e. the "empty/unnamed" parameter; see example).

Note that a shortcut function exists for `mimeEntity` objects, to obtain the header and parse it for parameters in one call; see `mimeEntityGetHeaderParameterValues` (p. 520).

EXAMPLE

```<$conType = 'text/html; charset="ISO-8859-1"; name="foo.html"'>
<$ret = (headerItemGetParameterValues($conType))>
Content type: $ret
<$ret = (headerItemGetParameterValues($conType, "charset" ))>
Charset: $ret
```

The output would be:

Content type: text/html
Charset: ISO-8859-1
CAVEATS

If the header being decoded is a potentially multi-item CSV list (e.g. “To” addresses), it should be split via headerGetItems (p. 550) first.

SEE ALSO

headerItemGetParameterNames, mimeEntityGetHeaderParameterValues
1.7.39 headerMailboxGetAddress

SYNOPSIS

varchar headerMailboxGetAddress(varchar mailbox)

Parameters:

- mailbox - A varchar mailbox (with address and/or display name)

Returns:

- The decoded varchar email address

DESCRIPTION

The headerMailboxGetAddress function returns the email address from a mailbox, with RFC 2047 encoded words decoded.

EXAMPLE

```php
$mailbox = "=?ISO-8859-1?Q?Troms=F8,?= Norway" <city@norway.org>
$ret = (headerMailboxGetAddress($mailbox))
Address: $ret
```

The output would be:

Address: city@norway.org

CAVEATS

When parsing a potentially multiple-mailbox header such as To, the value should be split into items first with headerGetItems (p. 550).

SEE ALSO

headerMailboxGetDisplayName
1.7. MIME API

1.7.40 headerMailboxGetDisplayName

SYNOPSIS

```sql
varchar headerMailboxGetDisplayName(varchar mailbox)
```

Parameters:

- `mailbox` - A varchar mailbox (with address and/or display name)

Returns:

- The decoded varchar display name

DESCRIPTION

The `headerMailboxGetDisplayName` function returns the display name from a mailbox, with RFC 2047 encoded words decoded.

EXAMPLE

```bash
$mailbox = "=?ISO-8859-1?Q?Troms=F8,?= Norway" <city@norway.org>
$ret = (headerMailboxGetDisplayName($mailbox))
Display name: $ret
```

The output would be:

Display name: Tromsø, Norway

CAVEATS

When parsing a potentially multiple-mailbox header such as To, the value should be split into items first with `headerGetItems` (p. 550).

SEE ALSO

`headerMailboxGetAddress`
1.7.41 mailParseAliases

SYNOPSIS

varstrlst mailParseAliases(varchar aliases, varchar what [, varchar flags])

Parameters:

- aliases - A varchar string in sendmail /etc/aliases format
- what - A varchar field indicating what to return:
  - "aliases" - List of entries’ aliases (left side)
  - "targets" - List of entries’ targets (right side)
  - "types" - List of entries’ types: "include", "fail", "pipe", "file", "address"
- flags - An optional CSV list of flags:
  - "includes" - Parse include files (default)
  - "noincludes" - Do not parse include files

Returns:

- A varstrlst list of aliases, targets or types

DESCRIPTION

The mailParseAliases function returns a list of aliases, their targets, or their types, as requested, from an /etc/aliases-style string buffer. The lists are parallel; thus a multi-target alias will be “denormalized” and cause that alias to be returned multiple times, once per target (see alias “foo” in the example).

By default (or if the “includes” flag is given), :include: targets will be read and processed as such. If the “noincludes” flag is given, includes will not be processed and will be returned as type “include”.

EXAMPLE

<capture>
# This is a comment.
postmaster: root
"foo": bar, joeblow@domain.com, 'joe blow'
nobody: fail: No such user
spam: /dev/null
orders: |/some/program

\(<\text{capture}>\)
\(<\text{aliasBuf} = \text{ret}>\)
\(<\text{aliases} = (\text{mailParseAliases(\text{aliasBuf, } "aliases" )))}>\)
\(<\text{types} = (\text{mailParseAliases(\text{aliasBuf, } "types" )))}>\)
\(<\text{targets} = (\text{mailParseAliases(\text{aliasBuf, } "targets" )))}>\)
\(<\text{loop } \text{aliases } \text{types } \text{targets}>\)
  Entry \text{aliases}: Type \text{types}, target is \text{targets}
\(<\text{/loop}>\)

The output would be:

Entry postmaster: Type address, target is root
Entry foo: Type address, target is bar
Entry foo: Type address, target is joeblow@domain.com
Entry foo: Type address, target is joe blow
Entry nobody: Type fail, target is No such user
Entry spam: Type file, target is /dev/null
Entry orders: Type pipe, target is /some/program
1.8 Vortex Library Modules

When creating a complex Vortex application, keeping the code in one large script can be unwieldy. Splitting it into multiple scripts is easier. But that usually means common code – such as look-and-feel or security functions – must be duplicated among the files. And when these functions are changed in one file, all the others must be updated too, which is time-consuming and error-prone. It may be unclear which files need to be updated without searching them for calls of common functions.

Keeping track of revisions is an issue also. One programmer may revise a script, unaware that another is still working on the same section, and stomp the other’s changes. A new version might introduce some bug, but without specific backups it’s difficult to fall back to a known safe version.

Library modules\(^{36}\) help solve these problems. Modules provide the following features to Vortex:

- **Modularity.** Code for a script can be split into logical sections instead of one big file.
- **Code sharing.** Functions can be utilized by multiple scripts without copying, while maintaining a single source point for the common code.
- **Dependency checks.** Scripts that share common code are automatically recompiled when that code changes, keeping them up-to-date.
- **Revision control.** Previous versions of code are directly available, as backups in case of errors or simply to “freeze” the code base for a given script.
- **Advisory locks.** A module can be locked to let others know that changes are in progress, helping prevent revision clashes.

A Vortex module is a script that is stored in the library and whose functions are callable by other scripts. The library is composed of the SQL tables SYSLIB and SYSDEP in the default database, which are automatically created as needed.

Every module has a name, by which other scripts can include it, and one or more dated revisions. New revisions of modules are inserted into the library by checking them in (the `-ci` command-line option). The source code can be retrieved for modification by checking out the module (the `-co` option). A module’s functions are included in a script for use with the `<USES>` directive.

1.8.1 Vortex Module Creation

A new revision of a library module is created by checking it in with the following command-line syntax:

```
```

The `-ci` option indicates check-in. The module will have the given name by which it is known to other scripts; if not specified it defaults to the base filename of the script. Its source code is taken from the

\(^{36}\)Library modules are available in Vortex version 2.6.936300000 19990902 and later.
script file. A log message describing the changes made can optionally be given with the -log message; if not specified the user is prompted for a message.

If the module already exists in the library, the last revision must be locked by the same remote host for check-in to occur. This helps ensure that only one programmer is working on the module at a time, so that changes are “serial” and seen by all. (The -force option can be given to override this check and force check-in regardless; the -CI combines -ci and -force.) A successful check-in unlocks the module by default; the lock can be maintained by giving the -lock option.

A new revision of the module with an incrementing version number is deposited. Any scripts that are known to use this module are flagged for re-compilation. In this way a script can be sure of always using the latest version of a module, without having to manually re-compile the script each time.

Note that a specific revision of a module is never modified once checked in: when the module is changed, a new revision is deposited. This preserves old versions in case functionality issues or bugs arise and a previous version is needed. Once checked in, a revision’s source file can be deleted if desired, as the source is stored in the library.

1.8.2 Vortex Module Revision

An existing module can be examined by checking it out. If it is to be edited as well, it should be locked to indicate this to other users (locked revisions are flagged in the output of -listlib.) Check-out occurs with this syntax:

texis -co [-module name] [-date D] [-rev N.N] [-lock] [script]

The module with the given name is checked out and written to the script file, or standard output if not specified. If the module name is not given, it defaults to the base filename of the script. If -lock is given, the module is locked as well. By default the latest revision of the module is checked out; a specific revision number can be examined with the -rev option. Or the latest revision on or before a (Texis-parseable) date D can be obtained with the -date option.

1.8.3 Using Modules in Scripts

Once created, a Vortex module’s functions can be utilized by other scripts or modules with a <USES> directive at the top of the script:

<USES [DATE=date] module[=file] [module[=file] ...]>

One or more modules are named in the directive. Each module’s code will be included when compiling the script. By default the latest revision of each module is used; if the DATE option is given then the latest revision on or before that date is used. This enables a script to “freeze” its usage in time and ignore later revisions, if for example the calling syntax for some functions in the module has changed, but the script hasn’t updated its usage.
Module functions are visible to the script according to their scope; see the discussion of function scope on p. 12 for details.

When any listed module is later changed, a script that `<USES>` the module – directly or indirectly – will be flagged for re-compilation on invocation. This ensures that scripts are kept up-to-date with respect to the modules they use, so that manual re-compilation is not needed.

For more information on the `<USES>` directive, see p. 69.

### 1.8.4 Vortex Library Option Summary

Several other command-line options exist for examining and maintaining the Vortex library. These options and the ones discussed above are summarized below:

- **-module M**
  Specify the name of a module. Required with most other library action options.

- **-ci**
  Check in a new revision of a module. Requires `-module`, optionally takes `-lock` or `-unlock` (the default) to lock or unlock the module as well. Optional `-force` forces check-in if no lock owned.

- **-CI**
  Shorthand for `-ci -force`.

- **-co**
  Check out (examine or modify) a module. Requires `-module`, optionally takes `-lock` or `-unlock` (the default) to lock or unlock the module as well. Optional `-rev` or `-date` to specify revision; default is latest.

- **-CO**
  Shorthand for `-co -force`.

- **-lock**
  Lock a module named with `-module`. Can also be used with `-ci/-co`.

- **-unlock**
  Unlock a module named with `-module`. Can also be used with `-ci/-co`.

- **-listlib or -ll**
  List the latest revision of each module in the library, when it was created, and if/when it was locked.

- **-LISTLIB or -LL**
  Same as `-listlib`, but verbose: list log messages too. Added in version 3.01.982000000 20010212.

- **-listrev or -lr**
  Like `-listlib`, but list every revision of each module. Optionally list revisions just for module named with `-module`. 
• **-LISTREV** or **-LR**
  Same as **-listrev**, but verbose: list log messages too. Added in version 3.01.982000000 20010212.

• **-wipelib**
  Completely erase the library and re-create its tables.

• **-del**
  Delete a module named with **-module**. Optionally delete only a specific revision with **-rev**, or all revisions older than a specific date with **-date**.

• **-targets**
  List targets – scripts and other modules – that depend on the module named with **-module**. This shows what scripts will be affected by a change in the module (new revision).

• **-diff**
  Compare two revisions of a module and print just the differences. Two **-rev** options specify the revisions. If the second is missing, the script file is compared; if the first is missing, the latest revision in the library is compared. **Note:** this option requires that the program **diff** (under Unix) or **fc** (under Windows) be in the path. The environment variable **DIFFCMD** may be set to an alternative program to use to compare files. Added in version 3.01.974500000 20001117.

• **-force**
  Force a lock/unlock of a module if not the owner of a lock. This should only be used when it is known that the lock owner is no longer working on the module.

• **-rev N.N** or **-rN.N**
  Give a specific module revision N.N for **-co, -del, -diff**.

• **-date D**
  Give a date specifying revision(s) checked in on or before date D for **-co, -del**.

• **-log msg**
  Specify log message **msg** when checking in a new module revision. The default is to prompt the user for a message.

• **-libdb db**
  Specify alternate database to store/read modules to/from; the default is the **texis.ini**-defined default database (p. 576). Used only for debugging/testing.

For other command-line options to Vortex, see p. 563 and p. 567.
1.9 Vortex Scheduling

Vortex scripts can be set up to run automatically on a given schedule. This is useful for periodic maintenance scripts, refreshing a Web site index, etc. The `<SCHEDULE>`\textsuperscript{37} directive schedules a script for automatic execution. For example:

```html
<SCHEDULE WHEN="daily at 12am">
```

Once the script is compiled or run so that this directive takes effect, the script will be run every day at midnight local time by the Texis Monitor in the background. The Texis Monitor itself (monitor) is automatically started whenever `texis` or `tsql` is run; to ensure it is always running for proper `<SCHEDULE>` operation, it is advisable to run it or another Texis program in the machine’s startup scripts at boot time \textit{as the Texis user}, not root. (Under Windows, the install program attempts to set this up automatically.) For more details on the `<SCHEDULE>` tag and its syntax, see p. 71.

1.9.1 Listing Scheduled Scripts

Currently scheduled scripts are noted in the SYSSCHEDULE table in the default database (usually `/usr/local/morph3/texis/testdb` or `c:\morph3\texis\testdb`). This table is created automatically when needed. To list schedules, the `-listsched` or `-ls` command-line option may be used. This option lists the currently scheduled scripts, their schedules, and when each is to be run next, in ascending order of next-run time. A typical listing of `texis -ls` might be:

<table>
<thead>
<tr>
<th>Next run</th>
<th>Schedule</th>
<th>Script</th>
</tr>
</thead>
<tbody>
<tr>
<td>in 26 sec</td>
<td>every 10 minutes</td>
<td><code>/usr/scripts/logchk</code></td>
</tr>
<tr>
<td>in 5h 10m</td>
<td>daily at 12am</td>
<td><code>/usr/scripts/walksite</code></td>
</tr>
<tr>
<td>in 8d 9h</td>
<td>monthly on first sun at 4am</td>
<td><code>/usr/scripts/logchk/monthly.txt</code></td>
</tr>
</tbody>
</table>

The `/usr/scripts/logchk` script is to be run every 10 minutes; it checks the most-recent logs for errors. Once a night at midnight a new Web site index is generated by the `/usr/scripts/walksite` script; this will be run next in 5 hours and 10 minutes. Finally, once a month on Sunday the `<monthly>` function of the `/usr/scripts/logchk` script is run to rotate Web server logs. An optional script name may be given after `-ls` to print information for just that script.

Alternatively, the `-LS` option prints schedule information with greater detail. Running this with an optional script name, e.g. `texis -LS /usr/scripts/logchk/monthly.txt`, we might see:

```
Script: /usr/scripts/logchk/monthly.txt
Schedule: monthly on first sun at 4am
Next run: 2004-04-01 04:00 (in 8d 9h)
Last run: 2004-03-04 04:00 (19d 15h ago) took 5m 13s with exit code 3
```

\textsuperscript{37}The `<SCHEDULE>` directive and associated options were added in Vortex version 3.01.985400000 20010323.
1.9. VORTEX SCHEDULING

First run: 2004-02-04 04:00 (47d 15h ago)
Status: runnable, 2 executions, last istag, id 3ab671c8107
Comments: Rotates logs once a month

This also tells us the last time the script was run, how long it took, that it exited with an error code of 3, and that its purpose is to rotate error logs.

1.9.2 Unscheduling Scripts

To unschedule a script, remove the desired <SCHEDULE> tag and re-compile or re-run the script. Alternatively, all schedules for a script may be removed with -sched all-off and the script name; however any <SCHEDULE> tags may become active again if the script is compiled or run.

All schedules for all scripts can be removed by wiping the schedule table with the -wipesched option. Be careful when using this option as every script’s schedule(s) will be removed.

1.9.3 Schedule Option Summary

The aforementioned command-line options and others that deal with scheduling in Vortex are summarized below:

- **-listsched or -ls**
  Summarizes currently-scheduled scripts.

- **-LS**
  Print current schedule information in greater detail.

- **-noeng**
  Do not translate vCalendar-syntax schedules to pseudo-English when listing. Normally this is done for readability, but the actual internally-stored syntax may be printed with this option.

- **-sched schedule**
  Schedule the script given on the command line with the given schedule. This schedule is distinct from any given by <SCHEDULE> tags in the script. schedule may be “off” to unschedule a previous -sched for this script, or “all-off” to unschedule both -sched and <SCHEDULE> tags for this script.

- **-start date**
  Specify a START date for the given -sched schedule.

- **-wipesched**
  Wipe the schedule table: remove all schedules for all scripts.

For other command-line options to Vortex, see p. 560 and p. 567.
1.10 Creating User Functions

To expand the functionality of Vortex, user-defined functions written in C can be added. User functions have a calling syntax in Vortex similar to built-in functions (which differ from script function calls – arguments are not named), and like built-in functions their return value is assigned to the variable $ret$.

1.10.1 Declaration and Argument Syntax

A user function has the following C declaration syntax:

```c
#include <sys/types.h>
#include <stdlib.h>
#include "tstone.h"
#include "vortex.h"

char **
myfunc(arg1, arg2)
char * arg1[];
char * arg2[];
{
    char **ret;

    /* ... do work with args ... */
    ret = (char **)NULL; /* malloc value, or NULL for none */
    return(ret);
}
```

Each argument to a user function is passed as a NULL-terminated array of pointers to strings, from the corresponding Vortex argument variable. The user function must not modify any of its arguments in any way. After doing its work, the function returns a string list that will become the $ret$ variable’s value. This return value is a malloc’d, NULL-terminated array of malloc’d strings. A function may also return just NULL to indicate an empty (no values) return value. The returned list is owned by Vortex and will be freed as necessary; no further reference to it is permitted.

User functions with zero to ten parameters are declared as above. If the function takes more than ten parameters, or a variable number of parameters, its arguments are given in an array:

```c
char **
manyargfunc(argc, argv)
int argc;
char **argv[];
{
```

38Even attempting to undo changes, e.g. temporarily terminating an argument string but restoring it when done, is not permitted: arguments may potentially be in read-only memory.
The string array at `argv[0]` is the first argument; `argv[1]` is the second argument, etc. The `argv[]` array is NULL-terminated after the last argument, and contains `argc` arguments. Since variable-arg functions can have any number of arguments, it is important that they check the value of `argc` before indexing into the `argv` array, and not go past the end of it. Functions with a fixed number of arguments (first syntax) will be checked at script compile time to ensure they are always called with the right number of arguments.

1.10.2 I/O

If a user function needs to print output, it must use one of the following functions. The normal C functions that print to `stdout` such as `printf()`, `puts()`, etc. must not be used, as they will interfere with normal output and will not be redirected to `<EXEC>` or `<CAPTURE>` as needed. These functions do not automatically HTML-escape their output; the `%H` format code must explicitly be used with `VXpf()` for escapement.

- `VXpf(const char * fmt, ...)`
  Replacement for `printf()`; has the same syntax. Also understands the same extended codes that `<fmt>` does.

- `VXwrite(const char * buf, size_t len)`
  Prints `len` bytes of `buf` to the current output. Binary data is permitted.

1.10.3 Compiling and Linking

Once code has been written for user functions, Vortex must be made aware of them, by creating a new `texis` executable (this is usually a symbolic link to monitor). The functions are listed in the global array `VUserFuncs[]` in the `vufuncs.c` file. This array determines what user functions are available in Vortex, their names, how many arguments they take, and their return type. It is an array of `VUSERFUNC` structs (declared in `vortex.h`):

```c
typedef char ** VUFUNC ARGS((char * arg[], ...));
typedef char ** VA_VUFUNC ARGS((int argc, char ** argv[]));
typedef struct VUSERFUNC_tag
{
    char     *name; /* name of function */
    VUFUNC   *func; /* pointer to VUFUNC or VA_VUFUNC to call */
    int      nargs; /* number of expected args (-1 for varargs) */
    int      type; /* FTN_... type to cast return value(s) to */
} VUSERFUNC;
```
Each user function available in Vortex must have an entry in this array, which is terminated with a NULL entry. The name field is the name of the function. func is a pointer to the function, declared in either of the two syntaxes above. nargs is the number of arguments the function expects. If -1 is given, then the function can take a variable number of arguments, and func is expected to point to a VA_VUFUNC-style declaration instead of VUFUNC-style. Non-variable-argument functions with more than ten arguments must also be declared VA_VUFUNC.

The type field is a FTN___ type (SQL type) that the returned string value(s) will be cast to as $ret. The default is VUTFYPT_DEFAULT, which is string (varchar). Other possible values include FTN_LONG for integer, FTN_DATE for a date field, FTN_FLOAT for a floating-point value, etc. Note that the returned C data from the function is always an array of char * text strings; the type field indicates what these strings will be cast to by Vortex when assigned to $ret.

To add new user functions, edit the vufunc.c file and add entries for your functions. Note that there are already entries for standard Vortex functions, such as sum, apicp, etc.

Compile vufunc.c and your code, and then re-link texis with vufunc.o, your code, and the Vortex, Texis and system libs. An example makefile is in the /usr/local/morph3/api directory; a typical version is excerpted here:

```
# 'monitor' is the actual executable; 'texis' is symlinked to it:
monitor:  vufunc.o vufuncex.o
   $(CC) $(LDFLAGS) $(OBJDIR)/monitor.o vufunc.o vufuncex.o \ 
   -lvhttpd -lvortex -lcgi -lntexis -ltexis -lapi3 -lm -o $@

# 'texis' and 'tsql' are just symbolic links to 'monitor':
texis tsql:  monitor
   rm -f $@
   ln -s monitor $@

vufunc.o:  vufunc.c vufunc.h
   $(CC) $(CFLAGS) -c vufunc.c
vufuncex.o:  vufuncex.c
   $(CC) $(CFLAGS) -c vufuncex.c
```

OBJDIR must be set to the installed API directory, normally /usr/local/morph3/api. The order of libs is important: note that the -lvhttpd lib is first, and the system libs like -lm are last. On some platforms, additional system libs may be needed at the end, such as -lsocket, and/or the Thunderstone libs may need to be included again at the end to resolve all symbols.

Note: Before running the new texis, old Vortex P-code files (*.vtx) must be deleted to ensure that scripts are re-compiled properly. Also, the executable or symlink must start with the name texis in order to run. Make sure that the proper path to the new texis is given when testing, e.g. ./texis, to ensure the old/current texis in /usr/local/morph3/bin isn’t run inadvertently to confuse things.

Once the new texis is tested, the new monitor that it points to can replace the stock monitor executable installed in /usr/local/morph3/bin (after backing it up). A symlink from texis to monitor should already exist there.
1.11 Vortex Command Line Options

Vortex can be run from the command line, for running of a script without a Web server or for maintenance scripts. The last argument is usually either a script to run or a SQL command to execute.

The Vortex command line syntax is:

```
texis [options] [var=value] [var=value] [srcfile-or-SQL-command]
```

(The brackets ([ ]) indicate an optional part of the syntax.) `srcfile-or-SQL-command` is either the full path of the Vortex script to run, or a SQL command to execute on the database. Either is required, unless a standalone action like `-W` is invoked.

If a script is named, it has the same syntax as the URL would have, without the CGI program (e.g. `/path[/+state][/function.mime][/+userpath]`). The path is interpreted as an ordinary file, not relative to `ScriptRoot` or document root. Note that running a script from the command line may result in inaccurate URLs begin generated by the `$url` and `$urlroot` variables, as the `texis` CGI script path cannot be determined without the server.

Variables may be assigned on the command line for a script with the `var=value` syntax. Both the `var` name and its `value` should be URL-encoded (for portable command-line escapement). Variable assignments may be intermixed with other options, but like all options, must occur before the script name which is last on the command line. Assigning the same variable multiple times gives it multiple values, in command-line order.

If a SQL command is given, the database defaults to the current directory, or the one named with `-d` is used. The result rows are printed out, in either columns (the default) or one field per row (if `-c` given). No Vortex variables may appear in the command; any parameters must be given as literal (single-quoted) values.

In addition to the following options, see also the library options (p. 560) and schedule options (p. 563).

### 1.11.1 Vortex Global Command-line Options

These Vortex command-line options apply to all actions.

- `-install-dir=dir`
  Use alternate Texis install `dir` instead of default or install-time-configured directory (e.g. `/usr/local/morph3` or `c:\morph3`). Added in version 5.01.1145307295 20060417.

- `-install-dir-force=dir`
  Same as `-install-dir=dir`, but try to use `dir` even if it does not exist. Added in version 5.01.1145307295 20060417.

- `-texis-conf=file`
  Use alternate Texis configuration `file` instead of `texis.ini` in the install directory. Added in version 5.01.1145307295 20060417.
1.11.2 Vortex Action Command-line Options

These Vortex command-line options specify an action to do; they are mutually-exclusive. See also the library and schedule options, however:

- `-?` or `-H`
  Print a help message.

- `--apply-license=|--}`file
  Apply an arbitrary Texis License update file (provided by Thunderstone, in `license.upd` format) directly. A user and password for authentication must be provided, with `-u` and `-p`. The file may be “-” (a dash) to indicate that standard input should be read; this can be useful for cut-and-pasting a license update to a command-prompt window. A success or failure message and exit code is generated. Note that license updates must be enabled via `texis.ini` (usually done at install); see the [License Update] settings in the Texis manual. Also see `<vxcp applylicense>`, p. 288. Added in version 6.

- `-C`
  Compile script only; do not execute. It is good practice to also use the `--warnings-are-fatal` flag with `-C` if running in a development (non-production) environment.

- `-K pid [sig]`
  Kill Texis process `pid`. An optional signal `sig` (name or value) may be given; the default is `SIGTERM`, which is a safer way to terminate a Vortex script (especially under Windows) than `SIGKILL` or Task Manager's End Process button. Under Windows, `SIGTERM` is mapped to a Vortex-specific soft-terminate event, and `SIGINT/SIGBREAK` are also mapped to soft events. Added in version 5.01.1172007224 20070220.

- `-license`
  Print current license statistics.

- `-License`
  Print more verbose license statistics.

- `-r`
  Run Vortex script. This is the default action, unless a SQL command is given.

- `-R script`
  Run Vortex script, specified in next argument, and pass remaining arguments to script via `<vxinfo scriptargs>` (p. 294), i.e. do not parse them as Vortex options. This is useful for Unix “shebang” scripts, as the last option on the “#!” command line, so that arguments to the script (not Vortex) may be passed in without flummoxing Vortex command-line syntax (which normally requires the script to be the last Vortex argument). See `--shebang example` (p. 570). Added in version 7.01.1395172000 20140318.

- `-s`
  Execute SQL statement. The remaining options only apply to SQL statements:
• **-update**
  Signal the Texis Monitor to try to update the Texis license, either by reading the "license.upd" file in the install dir (provided by Thunderstone), or by contacting the Thunderstone license server over the Internet.

• **-version**
  Print version information and exit.

• **-platform**
  Print just platform ID and exit. Added in version 6.00.12778 20100629.

• **-W**
  Wipe the state table (delete and re-create it). A srcfile need not be given. Occasionally running texis with this option when the server is inactive is recommended, as the state table may grow large on heavily used servers.

### 1.11.3 Vortex Optional Command-line Options

These Vortex command-line options modify other actions:

• **-d db**
  Use database db instead of the default-installed database path. A run-time `<DB>` statement in a script will override this.

• **-u user**
  Login as the given Texis user.

• **-p password**
  Login with the given password.

• **var=value**
  Assign the URL-encoded value to var on startup. Multiple assignment statements for the same variable will result in multiple values for that variable. Vortex scripts only.

• **-M**
  Output the source file:line first in error messages. This is useful in an editing/compilation environment such as Gnu emacs that can parse this output and automatically go to the source line of the error.

• **-e file**
  Log errors to file instead of the default/installed log file (/usr/local/morph3/texis/vortex.log). A - (dash) means log to standard error. Errors are also printed in the output, unless a putmsg function (p. 581) is defined to catch them.

• **-t n**
  Use a timeout of n seconds when running a script; -1 for no timeout. Overrides the script <timeout> (p. 59, but not run-time timeouts via <vxcp timeout>).
• **--shebang{{[, ]}options}**
  Parse comma- or space-separated Vortex options that are part of this same argument. This may be used in Unix “shebang” scripts, where most platforms either merge space-separated arguments on the “#!/” command line (after the interpreter) into one argument, or silently drop arguments after the first (e.g. Solaris), either of which would cause problems. By using **--shebang**, such a merged argument may be passed to Vortex such that it knows to space-separate the remainder of the argument into further options.

  Combining this option with **--R** enables a Vortex script to be made into a self-executing script (in Unix), and still pass Vortex options on the command line. E.g. to set the error log to standard error:

  ```
  #!/usr/local/morph3/bin/texis --shebang,-e,-,-R
  <script language="vortex">
  ...
  ```

  Note that **--R** is always last. The script (not Vortex) command-line arguments may be obtained with `<vxinfo scriptargs>` (p. 294).

• **--ignore-env-script-name=value,...**
  Override texis.ini setting [Texis] Ignore Env Script Name (p. 577). Usually the texis.ini setting is enough, but this command-line override can be used (e.g. in the web server config) if multiple CGI environments are in simultaneous use and the setting must differ between them. Note that it will be ignored in CGI mode (like all command line options) unless permitted via [Texis] Allow Cgi Command Line Options (p. 578). Added in version 5.01.1182304953 20070619.

• **--pause-at-exit**
  At program exit, pause until user presses Enter. Can be useful when running a script in a dedicated command prompt window, to avoid having the window close immediately after the script ends without having a chance to see its output. Added in version 5.01.1225327000 20081029.

• **--warn-if-ret-loop**
  Warn during compilation if \$ret is used in a loop context (without a ROW flag), either directly (e.g. `<loop \$ret>`) or indirectly (e.g. `<stat nonempty ...>`). This usage can cause problems for code called inside such a loop, because like any other variable, \$ret cannot have its type nor number of values changed while being looped over. Thus any new \$ret returned by calls inside the loop will be forced to have its type cast to the looped \$ret type, will lose all but the first returned value, and the first value will be placed in the middle of \$ret (i.e. the current loop value). This may result in loss of data, error messages from type conversions that fail, and unexpected return type(s).

  Use **--no-warn-if-ret-loop** to turn off this warning. The default is the value of the texis.ini setting [Texis] Warn If Ret Loop (yes/no); if that is unset, it defaults to “no”. Added in version 7.01.1409098000 20140826.

• **--warnings-are-fatal**
  Treat compilation warnings as fatal, i.e. fail the compile and exit non-zero. Without this flag set, some compilation messages are non-fatal if the compiler considers them benign (e.g. too few/many arguments for `<fmt>`); this may allow the script to finish compiling and run. In a production
environment (e.g. automatic re-compilation during a script upgrade), running the script – even with potential errors – may be more important than halting execution to fix a warning in code that might not even be reached at runtime; thus this flag is off by default. During development however, it is good practice to set this flag, to catch such errors early when stopping compilation/execution is more permissible.

Use `--no-warnings-are-fatal` to turn off this setting. The default value is the value of the `texis.ini` setting `[Texis] Warnings Are Fatal (yes/no); if that is unset, it defaults to “no”. Added in version 6.00.1306875000 20110531.

- **--var-arg-decode**
  URL-decode command-line variable assignment arguments (both variable name and value). This is the default. Added in version 7.00.1362529000 20130305.

- **--no-var-arg-decode**
  Do not URL-decode command-line variable assignment arguments, i.e. take the variable name and value as-is. Added in version 7.00.1362529000 20130305.

- **--compatibility-version{=| }version**
  Sets Texis compatibility version to given version. See `<vxcp compatibilityversion>` (p. 289) for details.

- **--lib-path{=| }path**
  Sets path to use when searching for Texis/Vortex shared libraries. See `<vxcp libpath>` (p. 287), which can override this at run time, for details. Added in version 7.02.1408567000 20140820.

- **--**
  Indicates end of options, i.e. what follows (if anything) is the script or SQL statement. Added in version 5.01.1182193063 20070618.

### 1.11.4 Vortex Command-line SQL Options

These Vortex command-line options are in effect when running a SQL statement from the `texis` command-line (the `--` option).

- **-c**
  Format one field per line in the output. Normally all fields from a row are output on the same line.

- **-h**
  Do not print column headings in the output.

- **-l rows**
  Limit output to the given number of `rows`.

- **-m**
  Create the database named with `-d`.

- **-v**
 Verbose: display inserted/updated/deleted rows also; by default only rows from `select` statements are shown.
- **-w width**
  Make field headings *width* characters wide. A value of 0 means align to the longest field name.

### 1.11.5 Vortex Miscellaneous Command-line Options

These Vortex command-line options are primarily for tracing and debugging.

- **-x [N]**
  Trap signals according to bit flags in integer N. Overrides `<TRAP>` directive (p. 79). The default trap value is 0x83. The default for N if unspecified is 0.

- **-tracesql [n]**
  Debugging: print informational, `<putmsg>`-capturable messages giving the exact SQL statements executed and their parameters. This is useful when debugging a complex script that constructs its SQL statements on the fly from various sources. Added in version 2.6.936400000 19990903. See also the `sqlcp` function (p. 126), and the `<TRACESQL>` directive (p. 76) for details on what values can be set. Specify multiple times for increased verbosity; in version 3.01.980200000 20010122 and later, an optional numeric argument *n* may be given instead.

- **-tracekdbf [file|n]**
  Debugging: trace KDBF calls. Takes an optional file or numeric argument. Unsupported/internal use. Added in version 3.01.983500000 20010301.

- **-traceskt [n], -tracedns [n], -traceidx [n]**
  Debugging: trace socket, DNS or Metamorph index calls. Use multiple times for increased verbosity. For tech support. See the same-name settings in the `sqlcp` and `urlcp` (p. 198) functions for details. Added in version 3.0.948700000 20000124.

- **-tracepipe n**
  Debugging: trace pipe calls. See the same-name setting under `vxcp` (p. 284) for details.

- **-tracealarm n**
  Debugging: trace alarm calls. Added in version 5.01.1106770783 20050126. Value is a bitwise OR of any of the following flags:
  - 0x0001: `TXsetalarm()`, `TXunsetalarm()`, `TXunsetallalarms()` called
  - 0x0002: System alarm/handler set/unset

- **-tracelib n**
  Debugging: trace shared lib calls. Flags subject to change in a future release. Currently defined values (bit-wise ORed):
  - 0x1: Trace libpath expansion
  - 0x2: Trace library search and loading
  - 0x4: Trace symbol lookup
  Added in version 5.01.1215040000 20080702.
1.11. VORTEX COMMAND LINE OPTIONS

- 0x0004: TX... alarms fired
- 0x0008: System alarms fired
- 0x0010: TXsetalarm(), TXunsetalarm(), TXunsetallalarms() finished
- 0x0020: Timestamp each message
- 0x0040: TXgetalarm() called

Values may change or be added to in future releases.

• -time
  Print user, system and real time used to standard error upon exit. Useful for timing a script or SQL statement.

• -conf file
  Deprecated option; use --texis-conf=file instead. Added in version 3.01.979600000 20010115.

• -mkwebinator
  Create Webinator tables and indexes. Unsupported/internal use. Added in version 3.01.994500000 20010706.

There are also several options associated with Vortex modules (p. 560) and scheduling (p. 563).

1.11.6 Differences Between Vortex, tsql and Metamorph API

While both the Vortex texis executable and tsql can execute SQL statements on the command line with similar syntaxes, there are subtle differences between the two. The texis executable is geared towards production servers, and restricts certain CPU-intensive operations. The tsql command is intended for temporary, administrative and debugging use, however for consistency with Vortex, in version 6 and later it has the same restrictions as Vortex on potentially high-load queries. (In version 5 and earlier, tsql was more lax; those legacy defaults can be restored by setting the SQL property querysettings to 'texis5defaults'; see p. 116.) Some other differences in default settings are listed in the following table:
<table>
<thead>
<tr>
<th>Behavior</th>
<th>Appliance</th>
<th>Vortex</th>
<th>tsql</th>
<th>MM3 API</th>
</tr>
</thead>
<tbody>
<tr>
<td>alipostproc (Post-processing)</td>
<td>off</td>
<td>off</td>
<td>off</td>
<td>–</td>
</tr>
<tr>
<td>allinear (Linear search)</td>
<td>off</td>
<td>off</td>
<td>off</td>
<td>–</td>
</tr>
<tr>
<td>alwithin (w/ operator)</td>
<td>off</td>
<td>off</td>
<td>off</td>
<td>on</td>
</tr>
<tr>
<td>alintersects (@N operator)</td>
<td>off</td>
<td>off</td>
<td>off</td>
<td>on</td>
</tr>
<tr>
<td>alequivs (Allow equivs: ` or keepeqvs)</td>
<td>off</td>
<td>off</td>
<td>off</td>
<td>on</td>
</tr>
<tr>
<td>allineardict (linear dictionary search)</td>
<td>off</td>
<td>off</td>
<td>off</td>
<td>–</td>
</tr>
<tr>
<td>exactphrase (Resolve noise in phrases)</td>
<td>off</td>
<td>off</td>
<td>off</td>
<td>–</td>
</tr>
<tr>
<td>qminwordlen (Min. query word length)</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>qminprelen (Min. wildcard prefix length)</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>qmaxsetwords (Max. words per query set)</td>
<td>500</td>
<td>500</td>
<td>500</td>
<td>no limit</td>
</tr>
<tr>
<td>qmaxwords (Max. search words in query)</td>
<td>1100</td>
<td>1100</td>
<td>1100</td>
<td>no limit</td>
</tr>
<tr>
<td>prefixproc (Prefix processing)</td>
<td>off</td>
<td>off</td>
<td>off</td>
<td>on</td>
</tr>
<tr>
<td>keepnoise (Keep noise query terms)</td>
<td>off</td>
<td>off</td>
<td>off</td>
<td>on</td>
</tr>
<tr>
<td>keepeqvs (Search for equivalences)</td>
<td>off</td>
<td>off</td>
<td>off</td>
<td>on</td>
</tr>
<tr>
<td>sdexp (Start-delimiter expression)</td>
<td>none</td>
<td>none</td>
<td>none</td>
<td>sent</td>
</tr>
<tr>
<td>edexp (End-delimiter expression)</td>
<td>none</td>
<td>none</td>
<td>none</td>
<td>sent</td>
</tr>
<tr>
<td>minwordlen (Min. morpheme proc. wordlen)</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>5</td>
</tr>
<tr>
<td>likepallmatch (Require all LIKEP words)</td>
<td>on</td>
<td>off</td>
<td>off</td>
<td>–</td>
</tr>
<tr>
<td>Multiple statements supported</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>–</td>
</tr>
<tr>
<td>Statement executed before column headers printed</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>–</td>
</tr>
</tbody>
</table>
1.12 Vortex texis.ini Configuration Settings

Certain site-wide properties of Vortex can be set in the Texis configuration file texis.ini\(^{39}\). This plain ASCII text file resides in the install directory (usually “/usr/local/morph3” under Unix, or “c:\morph3” under Windows). The format of this file is similar to Windows .ini files, where a section is denoted by a name in brackets on one line, and settings for that section follow, one per line:

```
[Section One]
Setting One = value
Setting Two = value

[Section Two]
Setting One = value
Setting Two = value
```

Setting and section names are case- and space-insensitive. A line starting with “;” (semicolon) or “#” (pound-sign) denotes a comment. For more details on other sections’ settings that apply to other programs, see the “Texis Configuration File” section of the Texis manual. Also note that many of these settings can be overridden by the Texis Web Server (vhttpd) config file; see p. 587.

A setting value may contain any of the following special variable references, which will be replaced as noted:

- `%INSTALLDIR%`: The Texis installation directory.
- `%BINDIR%`: The Texis binaries (executables) directory, typically “c:\morph3” under Windows and “/usr/local/morph3/bin” under Unix.
- `%EXEDIR%`: The parent directory of the currently running executable (or the Texis binary dir, if the former cannot be determined)
- `%SCRIPTROOT%`: The value of the setting [Texis] Script Root. Not usable in that setting’s value. This variable was added in version 6.

Note that `%DOCUMENT_ROOT%` and `%SERVERROOT%` should only be used in Web-only settings, e.g. [Texis] Script Root, not in settings that may also be relevant to command-line invocations.

The following Vortex-specific settings appear under the [Texis] section of texis.ini:

- **Vortex Log**
  Sets the log file for errors. The default is texis/vortex.log in the install directory. When running the Texis Web Server (vhttpd), this value can be overridden in the vhttpd.conf file (p. 589).

---

\(^{39}\) In versions prior to 6, the configuration file was called texis.cnf instead of texis.ini. Version 6 will try to load it from the old location if it cannot be found at the new location.
• Default Database
  Sets the database that is used if no database is specified by the script. This is also the database used to store modules. The default is texis/testdb in the install directory.

• Default Script
  Sets the script that is executed if none is specified in the URL, when Vortex is invoked from the web. The default is texis/testdb/index in the install directory.

• ErrorScript
  Sets the Vortex script to run if the requested script cannot be started, when Vortex is invoked from the web. This script would be run if the requested script does not exist, cannot be compiled, or otherwise cannot be started normally. Any putmsg errors generated by the attempt to start the requested script are saved for capture by the ErrorScript’s <putmsg> function. This enables the ErrorScript to control what errors are seen by the user, even for errors that other scripts can’t capture. It can then take appropriate action such as generating a useful error page, redirecting to another page, notifying the site administrator, etc. Vortex is installed with the error script texis/scripts/errorscript configured as an example.

  Note that any errors generated by the requested script after it has started are capturable by that script’s <putmsg>, not the ErrorScript. Hence the main script’s <putmsg> function should be the first place to put error checking. ErrorScript is for fatal startup errors.

The ErrorScript value is a file path to a Vortex script, typically /usr/local/morph3/texis/scripts/errorscript. If a relative path is specified, it is taken as relative to the web server’s SERVER_ROOT directory. This allows multiple ErrorScripts to be specified across multiple web servers via the single texis.ini file. (Note that the web server must set the environment variable SERVER_ROOT for relative paths to work.)

When running the Texis Web Server (vhttpd), the ErrorScript value can be overridden in the vhttpd.conf file (p. 593). Added in version 4.00.1018000000 20020405.

• ErrorFile
  Sets the plain HTML file to send if ErrorScript is to be invoked but is unspecified or cannot be started. The string %errors% in the file is replaced with the text of any error messages. Note that this is a plain HTML file, intended as a last-ditch “fallback” if ErrorScript fails. Primary responsibility for handling errors should lie with the <putmsg> function in the main script first, then the ErrorScript script, then ErrorFile.

The ErrorFile value is a file path to a plain HTML file, typically /usr/local/morph3/texis/scripts/errorfile. If a relative path is specified, it is taken as relative to the web server’s SERVER_ROOT directory. This allows multiple ErrorFiles to be specified across multiple web servers via the single texis.ini file. (Note that the web server must set the environment variable SERVER_ROOT for relative paths to work.)

When running the Texis Web Server (vhttpd), the ErrorFile value can be overridden in the vhttpd.conf file (p. 594). Added in version 4.00.1018000000 20020405.

• ScriptRoot
  Sets the root file directory to look for Vortex scripts, when run from the web. In old versions (4 and earlier) the default is the web server’s document root (i.e. the DOCUMENT_ROOT environment variable).
In version 5 and later, the default is the texis/scripts subdirectory of the install dir: this helps avoid permission problems with compiling scripts, e.g. where the CGI user does not have permission to write .vtx files to document root. It also prevents Web users from downloading the contents of Vortex scripts, as they are now outside the web server tree.

The ScriptRoot value is a file directory. If a relative path is specified, it is taken as relative to the web server’s SERVER_ROOT directory. This allows multiple ScriptRoots to be specified across multiple web servers via the single texis.ini file. (Note that the web server must set the environment variable SERVER_ROOT for relative paths to work.)

In version 5 and later, ScriptRoot can also be set to the special value “%DOCUMENT_ROOT%” (no prefixes/suffixes) to use the current document root as the script root. This may be for back compatibility with a version 4 installation, or where multiple web servers are being run and a different script root is desired for each server.

When running the Texis Web Server (vhttpd), the ScriptRoot value can be overridden in the vhttpd.conf file (p. 588).

This setting was added in version 4.00.1017300000 20020327.

- **Vortex Source Extensions**
  Space-separated list of file extension(s) for Vortex source files, including the period if applicable. Individual extensions may be quoted, e.g. if empty. The preferred/default extension is listed first. In version 6, the default is “.vs ’’”, i.e. the preferred extension is “.vs” but no-extension files are accepted for back-compatibility. In version 5 and earlier, the default was no extension. The source (or object i.e. “.vtx”) extension is optional when specifying a script to run via the command-line or web, except where a mapping requires it, i.e. [Httpd] Vortex By Ext Path, vhttpd VortexByExtPath or Apache CGI by-file-extension mappings. Added in version 5.01.1158816000 20060921.

- **Vortex Object File Access Method**
  Method to use to access Vortex object (.vtx) files; one of memorymap or read. The default is memorymap for Unix for speed, and read for Windows to allow re-compilation of in-use object files. Reading is slightly slower and uses more memory than memory-mapping, but in practice the difference is negligible. Added in Vortex version 7.01. Previous versions were always memorymap.

- **Ignore Env Script Name**
  When to ignore the SCRIPT_NAME environment variable when constructing the Vortex $url variable. In some CGI environments (particularly when the CGI mapping is by-file-extension and not by-directory), SCRIPT_NAME is set incorrectly or redundantly, and concatenating it with PATH_INFO (as per CGI/1.1 spec) would result in the wrong self-referential URL; in these instances SCRIPT_NAME should be ignored. The Ignore Env Script Name setting controls when this occurs. It is a comma-separated list of any of the following:
  
  - ext-IIS-prefix
    When SCRIPT_NAME ends in one of the non-empty Vortex Source Extensions or “.vtx”, and SERVER_SOFTWARE starts with “Microsoft/IIS”, and SCRIPT_NAME is a prefix of PATH_INFO. This typically indicates a Microsoft IIS/6.0 application mapping, where both SCRIPT_NAME and PATH_INFO contain the Vortex script.
  
  - redirect_handler=action-name
When the environment variable REQUEST_HANDLER is set to action-name. This may indicate an Apache 2.1+ Action/AddHandler by-file-extension mapping, where using SCRIPT_NAME would give a usable URL, but different from the request (and thus break cookies).

- exe-IIS-prefix
  When SCRIPT_NAME ends in “.exe”, and SERVER_SOFTWARE starts with “Microsoft/IIS”, and SCRIPT_NAME is a prefix of PATH_INFO. This typically indicates a Microsoft IIS/7.0 Script Map (instead of a CGI-exe map), where both SCRIPT_NAME and PATH_INFO contain the “texis.exe” value, and PATH_TRANSLATED is incorrect. SCRIPT_NAME will actually be removed from PATH_INFO instead of being ignored, in this instance. Added in version 5.01.1229472493 20081216.

- always
  Always ignore SCRIPT_NAME.

The default value is “ext-IIS-prefix, redirect_handler=texis-vortex”, which handles IIS/6.0, as well as Apache 2.1+ (if the Action is named “texis-vortex”). Added in version 5.01.1182304953 20070619.

- Allow Cgi Command Line Options
  Which Vortex command-line options to enable in CGI mode. This is a space-separated list of command-line options (each optionally double-quoted). The default is empty (none) for security, since the QUERY_STRING from the user may be mapped to the command line by the web server. However, it may contain the options --ignore-env-script-name or --, if for example the former must be set differently explicitly for different CGI environments (this is an unlikely situation). In such cases, -- (end of options) should also be allowed, and any server-configured command-line options should be configured and ended with -- before any user-supplied (QUERY_STRING) options are possible. Added in version 5.01.1182042673 20070616.

- Entropy Pipe
  The path to the prngd daemon’s Unix socket. This is only required for Unix systems, and only if using SSL (e.g, in <fetch>) and there is no random device (e.g. /dev/random). The default is %INSTALLDIR%/etc/egd-pool. Overridden by the entropypipe setting of urlcp (p. 212). Added in version 4.01.1031693207 20020910.

- Charset Converter
  The program and arguments to run for translating character sets (e.g. when <fetch>ing pages.) It should take stdin in one charset and output stdout in another requested charset. The variables %CHARSETFROM% and %CHARSETTO% will be replaced with the input and output (requested) charsets, respectively. Embedded-space arguments may be double-quoted. The default is "%INSTALLDIR%/etc/iconv" -f %CHARSETFROM% -t %CHARSETTO% -c. Added in version 5.00.1089408421 20040709.

- Charset Config
  The charset config file. See the charsetconfigfromfile setting of <urlcp> (p. 217) for details on the file format. If unset, the default charset config file is conf/charsets.conf in the install dir. If the config file cannot be found, an internal default config is used. Added in version 6.

- Default Header Printif
The default value for the PRINTIF flag to `<header>` statements. Overridden by setting PRINTIF non-empty in a `<header>` statement (p. 144). The default value is `headers`. Note that this default is a behavior change from prior versions (previous default was `always`). Added in version 5.01.1111422505 20050321.

- **Cookies**
  Whether to URL-decode incoming cookie values when initializing Vortex variables or not. The value `urldecode` means URL-decode the values, and is the default. The value `asis` means leave values as-is, i.e. do not decode. Can be overridden in a script with the `<COOKIES>` directive (p. 74). Added in version 5.01.1121884376 20050720 (default `urldecode` in previous versions).

- **Auto Create Db**
  If nonzero, automatically create a database if it does not exist when referenced, e.g. when used indirectly in an `<if>` or assignment. The default is 1. Added in version 5.01.1157523656 20060906.

- **CGI Debug**
  Controls whether to enable CGI debugging features, such as the `-dump` option, in CGI mode. Off (0) by default. Added in version 4.02.1047916696 20030317.

- **Default Vortex Sort Shortest**
  Nonzero: the default for `<sort>`/`<uniq>` should be `SHORTEST`, not `LONGEST`. Default is 0 (i.e. `LONGEST`). Added in version 5.01.1189552000 20070911. Previously all `<sort>`s were `SHORTEST`.

- **Array Convert**
  Sets default for `<sqlcp arrayconvert>`: same syntax. Default is on; can be set to off to restore pre-version-6 Vortex variable behavior with `strlst` etc.

- **Vortex Add Trailing Slash**
  Whether to add a trailing-slash (via a redirect) to “directory” script URLs – those without a function/MIME-extension or user-path. The default is no. See also the `<addtrailingslash>` directive (p. 81), which overrides this. Added in version 5.01.1227568000 20081124.

- **Exec Quote Args**
  Sets default for `<exec quoteargs>` flag (p. 45). Default is 1 (on). Added in version 6.

- **Texis Defaults Warning**
  Whether to issue a warning message when `<apicp texisdefaults>` is set, as in version 6 and later `texisdefaults` is deprecated, since Texis defaults have changed to match Vortex defaults. The `texisdefaults` setting still takes effect, but as it may be removed in a future release, a warning is issued to encourage its removal in scripts. The setting `<apicp querysettings texis5defaults>` should be used instead.

  The default for Texis Defaults Warning is on. However, if legacy scripts cannot be changed, the warning may be silenced by changing it to off. Added in Texis version 6.

- **Compile SQL Expressions**
  Whether to compile SQL expressions into the `.vtx` object file where possible, rather than interpret them on the fly at run-time. SQL expressions are used in parenthetical variable assignments (e.g. `<$x = ($y + 5)>`) and complex `<if>` statements, amongst other places. Compiling them speeds up
script execution, as the original expression then does not need to be re-interpreted by the SQL engine every time the statement is run.

The default is on, and can be overridden in a script with the `compilesqlexpressions pragma` (p. 82). Added in version 6.01.

- **Allowed Literalprint Prefix**
  When using `<!-- pragma literalprint off -->`, lines that begin with this prefix are still allowed to be printed. Useful for defining a project-wide debug syntax for quick output, but can easily be verified that it’s removed by compiling with this cleared.

  E.g. `Allowed Literalprint Prefix = +++` will allow lines starting with `+++`. The default is empty/unset/off.
1.13 Vortex Error Messages

Errors may occur any time Vortex is run, such as syntax errors during compilation, problems during execution, or incorrectly formed statements. These errors are logged with a timestamp to the error log file, which is set during installation or with the command-line option \(-e\) (p. 567). The default error log is /usr/local/morph3/texis/vortex.log or c:\morph3\texis\vortex.log under Windows\(^{40}\). The log file may be changed in the texis.ini configuration file (see p. 575). If the log file cannot be opened, errors are printed to standard error (stderr). Each line of the vortex.log has the following format:

NNN date script:line: Message in the function: dosomething

where NNN is a 3-digit code indicating the general type and severity of the error (not a unique identifier of the error). The first digit indicates the severity: 0 is an error, 1 is a warning, 2 and higher are informational messages. The other 2 digits indicate the type of error: 02-10 are file errors, 11 is out of memory, 15 is a usage (syntax) problem, etc. The date and time of the error come next, in YYYY-MM-DD HH:MM:SS format\(^{41}\). If applicable, the script name and line that the error occurred on are given. The rest of the line is a description of the error, possibly with the internal C function name where the error occurred.

Capturing Vortex Errors

If the Vortex function \(\text{putmsg}\) is defined in a script, it will be called when errors occur, instead of the errors being printed out and logged. The following variables are set when \(\text{putmsg}\) is called\(^{42}\):

- $errnum
  A number indicating the severity and type of the error. There are several classes of errors:
    - 000-099 Errors
    - 100-199 Warnings
    - 200-299 Informational messages
    - 300+ Miscellaneous/debug messages
  Note that the error number is to be used to classify the error (e.g. whether to display it or not); it does not uniquely identify the specific error message.

- $errscript
  The URL path of the Vortex script, or the module name in square brackets, where the error occurred.

- $errline
  The source line in the script or module where the error occurred.

- $errmsg
  A text message describing the error.

\(^{40}\)This may vary if Texis was installed in a different directory than /usr/local/morph3.

\(^{41}\)In version 4 and earlier, the format was MMM DD HH:MM:SS.

\(^{42}\)In a future version, these will be parameters to the \(\text{putmsg}\) function instead of global variables.
• $errfunc
  The C (not Vortex) function where the error occurred; empty ("") if unknown. This is useful for technical support.

• $errvfunc
  The Vortex function where the error occurred; empty if unknown.

These variables may have several values, if more than one error occurs at a time. The variables are reset each time putmsg is called.

This example putmsg function just saves all the messages until another function is called to print them out:

```html
<A NAME=putmsg PRIVATE>
  <$savnum = $savnum $errnum>
  <$savmsg = $savmsg $errmsg>
  <$savline = $savline $errline>
</A>
```

The following function can then be called at an appropriate point to print out the errors, such as at the end of the script, or after a <TABLE> has been completed:

```html
<A NAME=flushmsg PRIVATE>
  <LOOP $savnum $savmsg $savline>
    <SWITCH $savnum>
      <CASE lt 100>
        <B>Error</B>:<BR>
        <CASE lt 200>
          <B>Warning</B>:<BR>
          <CASE lt 300>
            Note:<BR>
            At line $savline: $savmsg <BR>
            </CASE>
        </CASE>
      </CASE>
    </SWITCH>
    <$savnum = >  <$savmsg = >  <$savline = >
  </LOOP>
</A>
```

By caching the messages, errors can be kept from breaking up the script’s desired HTML output, yet still be printed to the user if desired. Or certain errors can be ignored altogether, such as when fetching pages from a site that might be down.

**Changing Logging and Printing of Vortex Messages**

Defining a putmsg script function turns off logging and printing of error messages because it is assumed that the function will completely handle the error, logging it (or not) as the programmer sees fit. However, with the <PUTMSG> directive (p. 65) and the vxcp function (p. 284), logging and/or printing during a
putmsg function call can be re-enabled. This saves the hassle of writing a log function inside <putmsg> to save the errors. These calls can also be used to ignore messages only during certain sections of code (e.g. calling an error-prone but benign <EXEC> program) without having to write an explicit <putmsg> function to handle the errors:

...<vxcp putmsg all off><$sav = $ret>   <!-- ignore errors -->
<EXEC /usr/bin/my/buggy/program></EXEC>
<vxcp putmsg all $sav>   <!-- restore error handling -->
...

Avoiding putmsg Side Effects

It is important to note that the putmsg function can be called unexpectedly, so it should take care to avoid side effects. A typical case is the unexpected modification of $ret by putmsg after a statement generates errors. Variables like $ret and $loop should be preserved in putmsg with local variables if they are modified:

<A NAME=putmsg PRIVATE>
<LOCAL saveret=$ret saveloop=$loop savenext=$next>
  <LOOP $errmsg>
    <fmt "Note: %s\n" $errmsg>
  </LOOP>
  <$ret = $saveret>
  <$loop = $saveloop>
  <$next = $savenext>
</A>

Here the initial values of $ret, $loop and $next are preserved in local variables, and restored on exit, because they are modified by fmt and LOOP. In this way, these variables won’t be truncated after a function, if error messages cause a putmsg call.

Capturing Compile-time Error Messages

Note that some errors may be generated before the script begins running (e.g. compile warnings): in Vortex version 4.00.1018000000 20020405 and later these messages are saved until the script runs (if possible) and thus may cause putmsg to be called even before the normal start function. Programmers should thus call any header-HTML type function in putmsg as well as their start function (with a one-time-call check variable) to ensure such header HTML is printed before any putmsg output. E.g.:

<A NAME=look PRIVATE>
<!-- Our header HTML for every page -->
In this example, we call the \texttt{<look>} function to print our header HTML. We call it in any function that may start execution, in this case both the \texttt{<main>} function – for normal operations – and the \texttt{<putmsg>} function – in case errors occur before the script starts.

\section*{Fatal Vortex Error Messages}

Some errors are fatal and prevent the script from running at all. The most common are specifying a nonexistent script (i.e. an incorrect URL), or fatal compile errors. These errors can be captured by an ErrorScript specified in the texis.ini file. See p. 576 for details.

Note that a few severe errors, such as stack overflow, can occur after the script starts (so ErrorScript does not apply), yet cannot be captured by \texttt{<putmsg>} at all, because the script cannot continue execution to call \texttt{<putmsg>}.

\section*{Common Vortex Errors}

Some common error messages generated by Vortex include:

\subsection*{Can't open source file nnn: ...}

The Vortex script nnn, given in the URL or command line, could not be found or is unreadable. The file or one of its directory components might not exist (No such file or directory), or may be unreadable with Vortex’ user permissions. Keep in mind that many Web servers run CGI programs (such as Vortex) as a different user or permission level than normal programs.
1.13. VORTEX ERROR MESSAGES

Cannot open default script ...: ...
No Vortex script was given to execute and the default script index in the default database cannot be opened.

Function/variable must be alphanumerics
The function or variable name had illegal syntax. Names must be composed of alphanumeric characters, underscores, and periods.

Function ‘func’ already defined
The function “func” was already defined (as a user or builtin function, if indicated), and a redeclaration was attempted.

‘sometag’ is a reserved name (HTML/Vortex tag)
The token “sometag” is reserved (cannot be a function name) because it is an HTML or Vortex tag name. Function names cannot be HTML tags to avoid confusion when calling them.

Must have a function ‘main’ defined in script
All Vortex scripts must have a main function defined, as the default start point of the script. This error is often caused by a misspelled <SCRIPT LANGUAGE=vortex> tag: if the block is not recognized as Vortex, it is silently skipped (it’s assumed to be another script language).

No text permitted after variable in argument
Argument must be single variable or literal
Each argument to a function or Vortex statement must be a literal value (e.g. “hello”) or a single variable. Variables cannot be nested inside a string, e.g. “hello $there” is illegal as an argument. (The SQL statement is an exception; its embedded variables are SQL parameters.)

Variable expected
A variable argument was expected, not a literal.

Syntax error: ...
The syntax of a Vortex statement was incorrect, and should conform to the given syntax.

Wrong number of arguments to user function ‘func’ (expected nnn)
A user-defined or builtin function(func) was called with the wrong number of arguments.

Function ‘func’ has no parameter named ‘ppp’
A script function(func) was called with an incorrect parameter ppp.

Function ‘func’ requires parameter ‘ppp’
A required parameter ppp (one with no default value) was not specified in a call to function(func).

Function ‘main’ must be PUBLIC
The function main must always be PUBLIC, because it is the default start function to Vortex scripts.

Unterminated <IF>/</ELSE>/etc.

Syntax error: unterminated block
A block statement, such as <SQL>, <LOOP>, <EXEC>, or <SWITCH>, was not terminated with a corresponding closing (<\ntag>) tag.
A closing tag for a Vortex block statement was given where there is no such open statement.

**Unknown operator ‘op’**

“op” is not a valid Vortex operator (e.g. eq, lt, etc.).

**Format code ‘p’ illegal in format string**

The format code “p” is known but illegal in a \texttt{<fmt>} string.

**Too many/few arguments for \texttt{<fmt>} format**

Too many or too few arguments were passed to \texttt{<fmt>} for the given format string.

**State URL too long, truncated**

There were too many variables \texttt{EXPORTed} to the URL, and/or their values were too long to be contained in a 512-character URL. Some or all of the variables will be lost. A common mistake is to inadvertently leave an \texttt{EXPORT URL} variable set to a list of multiple values instead of one or none, or forgetting to clear it after a SQL loop: the next \texttt{$url} might export a lot of unneeded values.

**Stack overflow**

Too many commands or function calls were nested. This may be caused by recursive function calls (a function calling itself). There is a fixed limit of 250 nestable calls, to catch infinite recursion. See the \texttt{STACK} directive (p. 75) and the \texttt{vxcp} function (p. 284).

**Corrupt object file**

An inconsistency was detected in the Vortex object file (.vtx file). This can be caused by another program modifying the file, copying it incorrectly (e.g. FTP in non-binary mode), or copying an object file from a different operating system or environment (object files are optimized for each platform). The error can generally be corrected by deleting the .vtx file, as Vortex will then automatically re-compile the script source when it is run.
Chapter 2

Texis Web Server

2.1 vhttpd Overview

The vhttpd program is the Texis Web Script web server. It integrates a fast, compact web server with Texis Web Script. The vhttpd server can directly run scripts without having to execute a separate CGI program. It also has less resource overhead (memory and processes) per transaction than other servers. These factors allow it to improve the throughput of highly-hit web sites.

2.2 vhttpd Config File

On startup, the Texis Web Server server looks for a config file. This file is by default conf/vhttpd.conf in the directory configured at install time, usually /usr/local/morph3. The config file can be changed with the –f command-line option. It contains settings that affect the actions of the server. Its format is similar to Apache web server config files: settings are one per line, with space-separated values after the setting name. Blank lines and comments (#-sign to end of line) are ignored. For example, the following config file sets the TransferLog file, where transactions are logged:

```
# This is a comment. Set the transfer log:
TransferLog /usr/local/httpd/logs/transfer.log
```

In version 6 and later, config file values may contain the following %-variable references:

- %INSTALLDIR% - The Texis installation directory
- %BINDIR% - The Texis binaries (executables) directory
- %EXEDIR% - The parent directory of the server executable (or the Texis binary dir, if the former cannot be determined)
- %SCRIPTROOT% - The script root
Many file path settings are ServerRoot-relative. This means that if a relative path is given (one that does not start with a / or drive letter, after %-variable expansion), it is taken to be a subdirectory of ServerRoot. This allows an entire web server to be moved to another directory hierarchy by simply changing ServerRoot, if all other applicable paths are set relative to it.

A boolean setting is an on/off setting. A boolean value such as yes, no, on, off, 1 or 0 is expected.

### 2.2.1 `vhttpd` Log and Path Settings

These `vhttpd.conf` config file settings affect files used by the `vhttpd` server, such as where transactions are logged.

- **ServerRoot filepath**
  File path to the root of the server tree. `DocumentRoot`, the logs files, etc. are in subdirectories of this by default. Environment variables may be referenced in the path with $, and home directories with ~ (tilde). ServerRoot may be set to `%INSTALLDIR%` (the default) for the Texis installation directory. The server config file is by default `conf/vhttpd.conf` in this directory.

- **DocumentRoot filepath**
  File path to the root of the document tree. Web requests for documents will be served from here. For example, the URL `http://www.example.com/info/details.txt` would be served from the file `info/details.txt` in the `DocumentRoot` directory. ServerRoot-relative. The default is `htdocs`.

- **ScriptRoot filepath**
  File path to root directory of Vortex script tree. Overrides `texis.ini` value (see p. 576), which overrides default of `texis/scripts` in the install dir. ServerRoot-relative. This allows Vortex scripts to be stored in a location other than the server's document tree, e.g. for security or permission reasons. Added in version 4.00.1017300000 20020327.

  In version 5 and later, `ScriptRoot` can also be set to the special value “%DOCUMENT_ROOT%” (no prefixes/suffixes) to use the current document root as the script root. This may be for back compatibility with a version 4 installation.

- **LogDir dir**
  Directory for log and pid files; ServerRoot-relative. `TransferLog, ErrorLog, VortexLog` and `PidFile` are placed in here by default. If these are files with no dirs (the default), then a parallel web server can be set up with identical configuration, but logging elsewhere, by just giving a new `-logdir` on the command line. This eases setting up a quick test server – without mucking up the live server’s logs, or having to write a new `vhttpd.conf` file.

---

1This is similar to, but distinct from, any substitution the shell may do on command line arguments.

2In version 5 and earlier the `%INSTALLDIR%` token was not supported, though the default was still effectively the Texis installation directory.
Note that log file locations can be overridden per-transaction in directly-run Vortex scripts, via `<vxcp>` (p. 284), if the AllowLogFileOverride setting is true (the default); see p. 591.

- **TransferLog filepath**
  File path to transfer log. Each web hit will be logged to this file, which is in Common Log Format (CLF) unless modified by LogFormat. Depending on LogFormat and/or LogOptions settings, the User-Agent and Referer may be logged with each entry as well. If TransferLog is a file (no dir in its value), it is placed in LogDir. If a dir is given in the TransferLog path, it is then ServerRoot-relative. The default is transfer.log.

  Note that log file locations can be overridden per-transaction in directly-run Vortex scripts, via `<vxcp>` (p. 284), if the AllowLogFileOverride setting is true (the default); see p. 591.

- **LogOptions option [option ...]**
  Logging options. The option can be one or more of:

  - Referer to log Referer in the transfer log.
  - UserAgent to log User-Agent in the transfer log.
  - Combined to log both User-Agent and Referer in the transfer log.
  - pid to log the server subprocess PID.

  Note that LogFormat, if specified, overrides ‘LogOptions’.

- **LogFormat format**
  Specifies the format of transfer log entries. The format is the same `printf()`-like syntax used by Apache 2.4 in its LogFormat directive, with some codes added and some Apache codes unimplemented. See the Texis Monitor web server setting [Httpd] Log Format for details. The default format is Common Log Format, as modified by LogOptions. Added in version 7.01.1384230000 20131111.

- **ErrorLog filepath**
  File path to server error log. Server errors are logged here; these are distinct from Vortex errors which are logged to the VortexLog. If ErrorLog is a file (no dir in its value), it is placed in LogDir. If a dir is given in the ErrorLog path, it is then ServerRoot-relative. The default is error.log.

  Note that log file locations can be overridden per-transaction in directly-run Vortex scripts, via `<vxcp>` (p. 284), if the AllowLogFileOverride setting is true (the default); see p. 591.

- **VortexLog filepath**
  File path to Vortex log. Vortex errors from server-run (via VortexPath) scripts are logged here. Overrides texis.ini value, if any (p. 575. Note that general non-Vortex server errors are logged to the ErrorLog instead. (This config setting also does not affect where Vortex errors are logged for CGI-run Vortex scripts, which are run by the Vortex executable texis. That is only controlled by the texis.ini setting.)

  If VortexLog is a file (no dir in its value), it is placed in LogDir. If a dir is given in the VortexLog path, it is then ServerRoot-relative. The default is vortex.log.

  Note that log file locations can be overridden per-transaction in directly-run Vortex scripts, via `<vxcp>` (p. 284), if the AllowLogFileOverride setting is true (the default); see p. 591.
CHAPTER 2. TEXIS WEB SERVER

- **PidFile filepath**
  File where the process ID of the web server is written at startup. This can be killed by a server-shutdown script. If PidFile is a file (no dir in its value), it is placed in LogDir. If a dir is given in the PidFile path, it is then ServerRoot-relative. The default is vhttpd.pid.

- **TypesConfig filepath**
  File containing extension to MIME Content-Type mappings. This is a text file, commentable like the config file, with each line of the form:

  ```
  mime/type  ext [ext ...]
  ```

  where mime/type is a MIME Content-Type and ext is one or more filename extensions (no dot) to associate with the MIME type. This list is used to assign Content-Types for file downloads, based on the outermost file extension. It also sets the Content-Types for built-in VortexPath scripts, overriding the default list. The AddType config setting can also modify the MIME type list. TypesConfig is ServerRoot-relative; the default is conf/mime.types. If the file is missing, a default list is used, based on the default list for AllowExt. Extensions are case-insensitive. Added in version 3.0.949000000 20000127.

- **EncodingsConfig filepath**
  File containing extension to MIME Content-Encoding mappings. This is a text file, commentable like the config file, with each line of the form:

  ```
  mime/encoding  ext [ext ...]
  ```

  where mime/encoding is a MIME Content-Encoding and ext is one or more filename extensions (no dot) to associate with the encoding. This list is used to assign one or more Content-Encodings for each file download, based on non-outermost file extensions. The AddEncoding config setting can also modify this list. EncodingsConfig is ServerRoot-relative; the default is conf/mime.encodings. If the file is missing, a default list for the extensions gz, uu, and Z is used. Extensions are case-insensitive. Added in version 3.0.949000000 20000127.

2.2.2 vhttpd URL Settings

These vhttpd.conf config file settings modify how the vhttpd web server responds to various URLs.

- **AllowExt .ext [.ext ...]**
  Add the given extension(s) to the list of allowed file extensions. Only files with these extensions are accessible to users. If no AllowExt settings are given, the default list is:

  ```
  .au .bmp .css .doc .dvi .eps .gif .gz .htm .html .ief .jpe .jpeg .jpg .latex
  .mov .mpeg .mpg .pbm .pdf .png .ppm .ps .qt .ram .ras .rgb
  .rtf .snd .tex .texi .texinfo .tiff .txt .wbmp .wml .wmlc .wmlsc .xbm
  .xls .xml .xpm .xwd .Z .zip
  ```

  The default list is cleared by the first AllowExt setting. Later settings add to the previous settings. By default, Vortex scripts (no extension) and object files (.vtx extension) may not be downloaded.
by users, in version 2.6.928300000 19990602 and later, when AllowExt was added. A value of * (asterisk) will allow any extension. An empty string ("") will allow files with no extension, such as Vortex script source files.

- **AddAllowExt .ext [..ext ...]**
  Same as AllowExt, but does not clear default list when used for the first time, i.e. always adds to existing list. Added in version 5.01.1205376000 20080312.

- **ExcludeExt .ext [..ext ...]**
  Add the given extension(s) to the list of denied file extensions. The deny list is checked before the AllowExt allow list. The default deny list is empty (except in version 6 and 7, where it is no-extension, .vs and .vtx). Added in version 5.01.1205376000 20080312. In version 6.00.1276567000 20100614 and later, ExcludeExt none will clear the list.

- **AddExcludeExt .ext [..ext ...]**
  Same as ExcludeExt. Added in version 5.01.1205376000 20080312.

- **IgnoreCaseExt boolean**
  Whether to ignore case when checking file extensions. On by default in case-insensitive-file OSes (e.g. Windows). Added in version 5.01.1205376000 20080312.

- **AllowFileMask mask**
  Only allow access to files in DocumentRoot with at least one of these permission bits set. Note that files must still be accessible by User. The default is o=r, i.e. files must have other-read permission set to be accessible. Added in version 5.01.1147373000 20060511.

- **AllowDirMask**
  Only allow access to directories in DocumentRoot with at least one of these permission bits set. Note that directories must still be accessible by User. The default is o=r, i.e. directories must have other-read permission set to be accessible. Added in version 5.01.1147373000 20060511.

- **AllowLogFileOverride boolean**
  Whether to allow the error, transfer and/or Vortex log file settings to be overridden (changed) in Vortex, via <vxcp> (p. 284). The default is true. Can be set false to debug complex <vxcp> log file redirection issues. Added in version 5.01.1170123063 20070129.

- **AcceptMethods method [method ...]**
  List of HTTP methods to accept, e.g. beyond the implemented GET HEAD and POST methods, for Vortex scripts. GET HEAD and POST should be listed as well, if this setting is used. It is up to the Vortex script to implement other methods correctly. A single asterisk (“*”) may be listed to accept any. Added in version 5.01.1208373137 20080416.

- **Options [+|-]option [[+|-]option ...]**
  Set option flags. A “+” (plus) prefix turns on the option, a “-” (minus) turns off the option. Options can be:
  - **All**
    All options except for MultiViews. Added in version 5.01.1251269000 20090826.
  - **Indexes**
    Generate an automatic directory listing if no DirectoryIndex file exists for a directory.
- IndexesRobotsIndex
  Allow robots to index automatic directory listings, via `<meta name="robots">` tag.

- IndexesRobotsFollow
  Allow robots to follow links from automatic directory listings, via `<meta name="robots">` tag.

- LabelArgs
  Label processes’ args with remote IP address of client. This can help in determining which server sub-process is serving which client.

- MultiViews
  Allow content-negotiated variant files to be served. With this option enabled, if a requested file is not found as named, files with the same name but additional recognized file extensions (for MIME types and/or encodings) will be searched for. The files will be ranked according to the client's `Accept-*` header preferences, and the highest-ranked file will be served. Applies to implicit `DirectoryIndex` files too. For example, a request for “/dir/file” might return “/dir/file.html”, “/dir/file.txt.gz” etc. If variant(s) are found but are not deemed acceptable according to the client’s `Accept-*` headers, a 406 Not Acceptable response may result. Disabled by default. Added in version 5.01.1251269000 20090826. Currently, only the `Accept-Encoding` client header is respected.

- AddType mime/type ext [ext ...]
  Maps filename extension(s) ext to the MIME Content-Type mime/type for file downloads and VortexPath scripts. Overrides/adds to mapping list from `TypesConfig` file (p. 590). Added in version 3.0.949000000 20000127.

- DefaultType mime/type
  Sets default MIME Content-Type to use if undeterminable from file extension, for file downloads. Added in version 3.0.949000000 20000127.

- AddEncoding mime/encoding ext [ext ...]
  Maps filename extension(s) ext to the MIME Content-Encoding mime/encoding for file downloads. Overrides/adds to mapping list from `EncodingsConfig` file (p. 590). Added in version 3.0.949000000 20000127.

- VortexPath urlpath [script]
  Add the full-path (starts with `/`) urlpath as a server-run Vortex URL. Any URL request in this tree will be treated as a Vortex script and run by the server as such. For example, if VortexPath `/texis` is set, then a request for the URL http://www.example.com/texis/mydir/search will run the Vortex script mydir/search in the `ScriptRoot` directory. (Each matching path component must match fully however; e.g. the URL http://www.example.com/texisation/mydir/search would not run a script, given the same VortexPath.)

  To replace an existing Apache web server’s CGI-run Vortex with Texis Web Server-run Vortex, replace a config file entry such as this:

  ```
  ScriptAlias /cgi-bin /usr2/pub/httpd/cgi-bin
  ```
with an entry like this:

```
VortexPath /cgi-bin/texis
```

The optional `script` argument (ServerRoot-relative) was added in version 2.6.930850000 19990701. If it is given, then the script is `script` plus the remainder of the URL request (after `urlpath`). For example, if

VortexPath /texis/special /usr/local/specialscript/main.html/+ is set, then `http://www.example.com/texis/special/extra` will map to the script `/usr/local/specialscript, function main, with $userpath of “/extra”.

VortexPath may be given multiple times to map different URL paths. Depending on the current license, the word `texis` may be required in the URL path, to identify the URL as Vortex. Note that giving a `script` argument can result in non-standard behavior of `$url` and other variables, since part of the script path is not in the URL.

- **VortexByExtPath** `urlpath`
  Root URL path to allow Vortex scripts to be run by file extension. A request in this path with a “subdirectory” component that ends in one of the non-empty Vortex Source Extensions (texis.ini setting, p. 577) or .vtx will be run as a Vortex script. Typically set to `/`; e.g. the request “/dir/subdir/script.vs/func.html” would then run the script `dir/subdir/script.vs` in the `ScriptRoot` dir. Added in version 5.0.1182804000 20070625. Note that Vortex Source Extensions typically only contains non-empty values (e.g. .vs) in Version 6.

- **EntryScript** `filepath`
  File path to Vortex script to run before every web transaction. This can be used to check security permissions, ward off rogue users, etc. ServerRoot-relative. The default is none. In version 2.6.929600000 19990617 and later, if this script returns a non-zero exit code via `<exit>`, the actual request URL is not served and the transaction is abandoned. See also the ENTRYFUNC directive (p. 67) which is often more useful.

- **ExitScript** `filepath`
  File path to Vortex script to run after every web transaction. Generally used to log transactions to a database. ServerRoot-relative. The default is none. See also the EXITFUNC directive (p. 67) which is often more useful.

- **ErrorScript** `filepath`
  File path to Vortex script to run if requested script cannot be started. ServerRoot-relative. This script can process errors for nonexistent scripts, compile problems, etc. Overrides texis.ini setting (see p. 576). Added in version 4.00.101800000 20020405.

- **UseErrorScriptForServerErrors** boolean
  Whether to invoke ErrorScript for certain server errors, such as 404 Not Found; normally it is invoked only for Vortex errors. If enabled, the function `statusNNN.html` in ErrorScript is called, where `NNN` is the HTTP response code. The following variables are also set:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>$response</code></td>
<td>HTTP response code</td>
</tr>
<tr>
<td><code>$httpurl</code></td>
<td>URL of request</td>
</tr>
<tr>
<td><code>$url</code></td>
<td>Request URL, not including URL path</td>
</tr>
<tr>
<td><code>$method</code></td>
<td>Method of request</td>
</tr>
<tr>
<td><code>$userpath</code></td>
<td>URL path of request</td>
</tr>
<tr>
<td><code>$servername</code></td>
<td>Server name</td>
</tr>
<tr>
<td><code>$status</code></td>
<td>HTTP status code</td>
</tr>
<tr>
<td><code>$request</code></td>
<td>Full request string</td>
</tr>
<tr>
<td><code>$server</code></td>
<td>Server name and port number</td>
</tr>
</tbody>
</table>

These should be verified as being from the server, for security; see the default ErrorScript `<entry>` function.
CHAPTER 2. TEXIS WEB SERVER

- STATUS_CODE The HTTP response code, e.g. 404
- STATUS_REASON The HTTP status line reason, e.g. Not Found
- STATUS_MSG A verbose text error message

The ErrorScript can use these variables to distinguish Vortex script errors (where the variables are not set) from web server errors (where they are set), as well as issue the proper HTTP response. See the default ErrorScript for example usage. Added in version 5.01.1206660000 20080327.

- ErrorFile filepath
  File path to plain HTML file to send if ErrorScript is to be invoked but cannot be started. ServerRoot-relative. This is a plain file “fallback” for ErrorScript failure. Overrides texis.ini setting (see p. 576). Added in version 4.00.1018000000 20020405.

- ProxyScript filepath
  File path to Vortex script to run for a proxy request. ServerRoot-relative. Default is none (do not proxy). Setting this script enables the web server as a proxy server. This script is then responsible for fetching and serving all such proxy requests (whether Vortex scripts or file requests). Such a script could compress images, strip ads, etc. for users’ everyday web surfing, if users configure their browsers to use the server as a proxy server. Added in version 4.00.994800000 20010710.

- ScriptAlias urlpath filepath
  Alias urlpath to CGI directory or program filepath. A web request in the urlpath URL tree will cause the CGI program filepath to be run, if it is an executable file. If filepath is a directory, the CGI program in it is named by the next component in the URL. The filepath is ServerRoot-relative, and ScriptAlias may be used multiple times. In version 2.6.930850000 19990701 and earlier, the filepath could only be a directory.

  For example, given ScriptAlias /cgi-bin cgidir and the directory cgidir in ServerRoot, a URL such as http://www.example.com/cgi-bin/prog will execute the CGI program cgidir/prog.

2.2.3 vhttpd OS/Resource Settings

These vhttpd.conf settings control various operating-system options or resource limits.

- User user
  Sets the user to access files and run scripts/programs as. It is important that this be a non-root user to help prevent security violations such as reading protected files, etc., as the web server generally must be started as root (to bind to port 80). This should be the same low-permissions user that Texis was installed as, so databases can be accessed by Vortex URLs.

  In versions after 5.01.1170498000 20070203, the default is the special token %TEXISOWNER%, which means the owner of the texis executable in the install directory. This avoids the need to separately configure the User after setting the proper owner of Texis files at installation. %TEXISOWNER% will not allow the user to be root, if that is the texis owner for some reason. In previous versions, %TEXISOWNER% was not supported, and the default user was “nobody”.
2.2. VHTTPD CONFIG FILE

- **Group** group
  Set the group to access files and run scripts/programs as. The default is user’s group.

- **ServerName** hostname
  Set the hostname to report in the $SERVER_NAME environment variable to CGI and Vortex scripts. The default is the value reported by gethostname().

- **BindAddress** hostname
  Local hostname to listen for web requests on. If the local machine has several addresses, the server can be set to listen on only one address. The default (or * if given) is all local addresses.

- **Port** number
  TCP port number to listen on. The default is port 80.

- **Timeout** seconds
  Maximum time to read/write data from/to a client. The default is 300 seconds. (The timeout set with this setting does not apply to server-run Vortex scripts, where the script timeout is used instead.)

- **MaxBacklog** number
  Maximum pending-connections backlog to allow; further requests will be refused or ignored depending on the OS. The default is 1024 or the operating-system defined limit.

- **MaxClients** num
  Allow at most num simultaneous connections (clients) to the server. When the limit is exceeded, an error will be noted in the log, and further connections will be ignored until some current connection(s) are completed.

  The MaxClients setting allows a server to deal with an overload from too many connections in a controllable manner. Instead of allowing ever more connections and slowing further, possibly taking the machine down, it waits for resources to become available. The default value for MaxClients is 32. A proper value for a given configuration depends on numerous factors, such as the load from individual scripts, available RAM, the hardware, OS etc. It is generally determined from experimentation. Too low a value will unnecessarily delay clients and show errors in the log. Too high a value may not allow the server to recover from a heavy load or a malicious client.

- **MaxRequestHeaderSize** num
  Accept at most num total header bytes – request line plus request headers – from a client request. Exceeding this limit will log an error and return a 413 (Request Entity Too Large) error to the client. Added in version 7.00.1373329000 20130708. The default is 65536; previous versions were hard-coded to 4096. This limit is to help prevent broken or malicious clients from consuming too many server resources.

- **PassEnv** var [var ...]
  Pass the named environment variables from the server’s environment to CGI and VortexPath environments. Typically used to pass things like LD_LIBRARY_PATH for CGIs using shared libraries, since the default CGI environment is restricted. Use caution to avoid passing secure information to insecure scripts. Added in version 3.0.948500000 20000121.

- **SetEnv** var value
Set the given variable to the given value in the environment for CGI and server-run (VortexPath) scripts. Use caution not to pass secure information to insecure scripts, or overwrite standard CGI variables like PATH_INFO. Added in version 3.0.948500000 20000121.

- **UnsetEnv var [var ...]**
  Unset (remove) the given variable from the environment for CGI and VortexPath. Use caution not to delete standard CGI variables like PATH_INFO. Added in version 3.0.948500000 20000121.

- **BadContentLengthWorkAround int**
  If bit 0 is set, work around issue with some browsers that send extra unexpected data beyond Content-Length. This data is not normally read, so the server’s socket close at transaction end causes a connection-reset to the browser, and the browser displays an error page. The problem often appears on large POST-data pages. If bit 1 is set, log the event. The default is 1 (i.e. work around issue but do not log). Added in version 5.01.1159494272 20060928.

- **LogEmptyRequests boolean**
  If true, empty HTTP request lines from clients (zero bytes read) will be logged to the error log. The default is off, as these connections are usually benign unused connections from browsers. No transfer log entry can be made, as there is no HTTP request. Added in version 7.00.1350603750 20121018.

### 2.2.4 vhttpd Optimization Settings

Below are vhttpd.conf performance settings. None need be set by default, but careful tuning of these parameters can sometimes increase server performance.

- **OutPktSz bytes**
  Set the TCP data packet size assumed in output. Most platforms send data in constant-size packets where possible. The server can tune its output buffers to this packet size, reducing the number of packets sent. The default is 512. It is only advisable to change this setting if you know what the packet size is.

- **OutBufSz bytes**
  Set the output buffer size. Must be a multiple of OutPktSz. The default is 1024.

- **SingleUser boolean**
  Set Texis single-user mode. Use with caution; see the caveats on singleuser with the <sqlcp> function, p. 126.

- **TxPreOpenDb filepath**
  The database to pre-open Texis handles for. In conjunction with the other TxPreOpen... settings, some Texis handles can be pre-opened by the web server. This can speed up the servicing of server-run Vortex scripts that use that database, by avoiding the overhead of opening a Texis handle for every request. Generally 10-20 handles are opened with TxPreOpenNum, and one or two handles are reserved per process with TxPreOpenGive, depending on the number of simultaneous SQL accesses in the script. ServerRoot-relative. The default is none.

- **TxPreOpenNum num**
  Pre-open num Texis handles in the TxPreOpenDb database. The default is none (0).
2.3. VHTTPD COMMAND LINE OPTIONS

- **TxPreOpenGive num**
  Give num pre-opened Texit handles to each process. The default is 2 (if TxPreOpenDb and TxPreOpenNum are set).

- **TxPreOpenLife num**
  Use each pre-opened Texit handle num times before closing. The default is 10.

2.2.5 vhttpd Miscellaneous Settings

These are miscellaneous or debug vhttpd.conf config file settings. They are not typically used by live servers.

- **VortexObjExt .ext**
  Debug setting. Use the filename extension .ext for Vortex object files compiled or run by the server. The default is .vtx.

- **Trap Signals boolean|integer**
  Debug setting. Whether to trap signals, and (if integer bit flags set) what to do when they occur. The default is yes. **Note:** Fundamental server performance may be impaired if this setting is changed. Same bit flags as the <TRAP> directive in Vortex (p. 79); however not all bit flags are supported in the server process itself.

- **Trap Connreset boolean|integer**
  Debug setting. Whether to trap connection reset of stdout (e.g. browser user hitting stop button) in Vortex scripts. See also <TRAP> directive in Vortex (p. 79).

- **WatchDog boolean**
  Debug setting. Whether to use a watchdog process or not. The default is no.

- **DebugLevel num**
  Set the debug level:
  - 0 Normal operation, the default.
  - 1 Do not daemonize on startup (stay in foreground).
  - 2 Do not fork: service one request and exit.

2.3 vhttpd Command Line Options

The following command line options may be given to the vhttpd server. Where applicable, these options override the corresponding settings given in the config file. For example, an identical server can be set up on port 1234 with a log directory of /tmp/logs without changing the config file, with the command line vhttpd -p 1234 -logdir /tmp/logs.

- **-d dir**
  Set ServerRoot to directory dir.
• `-f file`
  Use the given server config file.

• `-k`
  Just kill (stop) the running `vhttpd` and exit.

• `--reinit`
  Just reinitialize the running `vhttpd` and exit. Currently the only reinitialization that occurs is that the `TransferLog`, `ErrorLog` and `VortexLog` configured at startup (or via `<vxcp>`) are re-opened; this releases the disk space of the original files if they were deleted for log rotation. If a log file cannot be re-opened (e.g. permissions), the previous file handle remains open. For log rotation, re-initializing the server via `--reinit` instead of killing and re-starting it avoids a potential few seconds of down time (i.e. connection-refused errors in browsers). Added in version 5.01.1170023637 20070128.

• `-version`
  Just print version information and exit.

• `-h`
  Print brief help message and exit.

• `-H`
  Print verbose help message and exit.

• `-doc dir`
  Set `DocumentRoot` to directory `dir`.

• `-p port`
  Set the TCP Port to listen for HTTP requests.

• `-V path`
  Add `path` as a `VortexPath` URL. May be used multiple times.

• `-s`
  Set Texis single-user mode. *Use with caution*; see the caveats on `singleuser` with the `<sqlcp>` function, p. 126.

• `-logdir dir`
  Set the `LogDir` to `dir`. (See `LogDir` above.)

• `-l file`
  Set the `TransferLog` to `file`. Can be overridden per-transaction with `<vxcp>` in Vortex scripts; see p. 284.

• `-ref`
  Log the HTTP Referer field in the transfer log. The `LogFormat` setting overrides this.

• `-ua`
  Log the HTTP User-Agent field in the transfer log. The `LogFormat` setting overrides this.

• `-e file`
  Set the server `ErrorLog` to `file`. Can be overridden per-transaction with `<vxcp>` in Vortex scripts; see p. 284.
2.4. VHTTPD CGI Environment Variables

The following environment variables are set by the Texis Web Server (vhttpd) when running CGI programs and Vortex scripts. These can be added, deleted or modified with the config file settings SetEnv, UnsetEnv and PassEnv.

- **PATH**
  Set to “/usr/local/bin:/usr/ucb:/bin:/usr/bin”.

- **-v file**
  Set the VortexLog to file. Can be overridden per-transaction with `<vxcp>` in Vortex scripts; see p. 284.

- **-P file**
  Set the PidFile to write the server process id to on startup.

- **-scriptroot dir**
  Sets the Vortex ScriptRoot dir.

- **-n num**
  Set TxPreOpenNum, the number of pre-opened Texis handles.

- **-db path**
  Set TxPreOpenDb, the database for pre-opened Texis handles.

- **-per num**
  Set TxPreOpenGive, the number of pre-opened handles to give each process.

- **-i num**
  Set TxPreOpenLife, the number of times to use a pre-opened handle.

- **-u user**
  Set the User to run web transactions as.

- **-g group**
  Set the Group to run web transactions as.

- **-b address**
  Set BindAddress, the local address to bind to.

- **-objext .ext**
  Debugging option. Set VortexObjExt, the filename extension for compiled Vortex object files.

- **-w**
  Debugging option. Use watchdog process.

- **-D**
  Set debug level. May be given multiple times.
• **GATEWAY_INTERFACE**
  The CGI interface supported by the server, “CGI/1.1”.

• **SERVER_SOFTWARE**
  The version/release of the server software.

• **SERVER_NAME**
  The local hostname, or the value given by the ServerName config command.

• **SERVER_PROTOCOL**
  The HTTP protocol used by the client to communicate with the server. Typically “HTTP/1.0”.

• **SERVER_PORT**
  The TCP port the server is running on, given by the Port config command or `-p` command line option. Usually 80.

• **SERVER_ROOT**
  The root filesystem directory where the server is based from. Given by the ServerRoot config command or the `-d` command-line option.

• **DOCUMENT_ROOT**
  The root filesystem directory where documents are served from. Given by the DocumentRoot config command or the `-doc` command line option.

• **REQUEST_METHOD**
  The HTTP method of the client’s request. Typically “GET” or “POST”.

• **REQUEST_PATH**
  The URL-decoded value of the request URL, without the query string. This is a non-standard variable.

• **REQUEST_URI**
  The original (non-decoded) request URL path and query string. This is a non-standard variable. Added in version 3.01.963872250 20000717.

• **PATH_INFO**
  The URL-decoded value of the remainder of the request URL, if anything is left after any prefix was mapped to a CGI program or VortexPath. Does not include the query string. PATH_INFO can vary from the actual URL if VortexPath aliases to a specific script.

• **PATH_TRANSLATED**
  Same as PATH_INFO, but translated to the filesystem, i.e. with DocumentRoot prepended.

• **SCRIPT_NAME**
  The decoded URL prefix path to the current CGI program or VortexPath.

• **QUERY_STRING**
  The request URL, not decoded, after the first question-mark (“?”).

• **REMOTE_ADDR**
  The dotted-decimal IP address of the client.
2.4. VHTTPD CGI ENVIRONMENT VARIABLES

- **REMOTE_HOST**
  Same as **REMOTE_ADDR**. (Client addresses are not currently reversed into hostnames to save load and network traffic.)

- **CONTENT_TYPE**
  The value of the `Content-Type` header, if any, given by the client.

- **CONTENT_LENGTH**
  The value of the `Content-Length` header, if any, given by the client.

In addition, any headers sent by the client are passed as an environment variable of the same name, in all upper case, with non-alphabetic characters mapped to underscore, and `HTTP_` prefixed. Typical examples include:

- **HTTP_REFERER**
  The value of the `Referer` header, if any.

- **HTTP_USER_AGENT**
  The value of the `User-Agent` header, if any.

- **HTTP_HOST**
  The value of the `Host` header, if any.

- **HTTP_ACCEPT**
  The value of the `Accept` header, if any.

- **HTTP_ACCEPT_ENCODING**
  The value of the `Accept-Encoding` header, if any.

- **HTTP_COOKIE**
  The value of the `Cookie` header, if any.

- **HTTP_CONNECTION**
  The value of the `Connection` header, if any.

---

4In version 3.01.984200000 20010309 and earlier, only certain headers were passed: the ones listed as examples.
Appendix A

SSL Client/Server Certificate Verification

The Vortex `<fetch>` library, and the Texis Monitor schedule/web servers, can verify peer (remote server or client) certificates. The Vortex `<urlcp sslverifyserver>` setting (p. 215) and the Texis Monitor schedule/web servers’ SSL Verify Client settings control this verification; they are normally “off”. When “on”, peer certificates are verified: if the verification fails, the SSL connection is terminated with a “Cannot verify certificate from host:port: reason at depth N” error. The reason can vary, as detailed in the error listings later.

A.1 Verification Process

SSL peer certificate verification in Texis/Vortex is handled by OpenSSL (except where noted below), and consists of a number of steps as listed here. For more details, see the OpenSSL `verify` command documentation, at `www.openssl.org/docs/`.

A.1.1 Certificate Presentation

First, a peer certificate must be presented. Servers generally always have a certificate to present, as it is required by SSL.\(^1\) Client certificates are optional in SSL however, so remote browsers (or remote Vortex clients) may have to have a certificate configured (e.g. via `sslcertificatefile`, p. 213). Note that if the `sslverifyserver` or SSL Verify Client setting is “optional” (or contains “-No_Peer_Certificate”), failure to present a certificate is not an error (though presented certificates still go through the checks that follow); this may reduce security however.

A.1.2 Chain Completion

Next, the peer certificate’s chain, or “pedigree” of issuer certificates, is established. The chain consists of the certificate’s issuer certificate (if any), followed by that issuer’s issuer certificate (if any), etc. up through a root or self-signed certificate. Such issuer certificates are generally CA (Certificate Authority) certificates,\(^1\)

---

\(^1\)Unless an anonymous cipher is used.
as opposed to the leaf (server or client) certificate itself, which is generally not a CA. A certificate’s chain may be provided by the peer itself (e.g. via the sslcertificatechain or SSL Certificate Chain File settings on the peer), and/or it may be automatically completed locally from trusted certificates. In any event, the chain is constructed and verified locally by looking for each chain certificate’s issuer certificate.

Finding Issuer Certificates

When looking for an issuer certificate, two possible sources may be consulted: trusted and untrusted certificates. Trusted certificates are ones the local client (Vortex) or local server (Texis Monitor) explicitly trusts: just the ones listed in sslecacertificatefile or SSL CA Certificate File. Untrusted certificates are ones obtained from the peer – the peer certificate and peer-provided chain certificates (if any).

Which of the sources are used depends on where the subject certificate (the one whose issuer is being looked up) was itself found. If the subject certificate was from the peer, then untrusted certificates will be searched first, and if the issuer is not found there, trusted certificates will then be searched. However, if the subject certificate was from the trusted certificates, only trusted certificates will be searched for the issuer. I.e. an untrusted (peer) certificate will never be used as an issuer of a trusted certificate.

This search difference can also result in slightly variant errors for a missing issuer certificate, depending on where the subject certificate is from. The error “unable to get issuer certificate” results if the subject certificate (whose issuer cannot be found) was a trusted certificate. However, “unable to get local issuer certificate” results if the subject certificate was an untrusted peer certificate. This message difference may be an aid in tracking down the error cause, as it enables the subject certificate’s source to be inferred\(^2\).

Note also that a peer certificate’s entire chain must be completed, through its root, for it to pass verification – regardless of whether the peer certificate itself, or an intermediate certificate in its chain, is trusted.

A.1.3 Extensions Check

Another check involves verifying that all untrusted certificates’ extensions are consistent with their stated purposes.

A.1.4 Root CA Trust

The chain’s root certificate is checked for trustworthiness: it must be a CA certificate (CA:TRUE in extensions), and it must be trusted locally (i.e. be listed in sslecacertificatefile or SSL CA Certificate File). Note that this means that if the peer certificate is self-signed (and thus a root certificate itself), it must also be a CA certificate; however, CA certificates are typically not used as server certificates and may cause a warning at server startup.

\(^2\)These errors and their causes are based on observed OpenSSL 0.9.7e behavior. Other versions, and OpenSSL documentation, may differ.
If the peer chain’s root certificate is not trusted, the “Cannot verify certificate ...” reason that results is usually “self signed certificate in certificate chain”.

A.1.5 Chain Expiration and Signature

The chain is also checked for validity: e.g. the current time must be within the Not Before and Not After validity ranges of all certificates, and signatures are checked.

A.1.6 Common Name Matches Hostname

Lastly, the peer certificate’s Common Name (CN) is checked against the hostname in the URL used to access the peer: if they do not match (case-insensitively), the verification fails. This check is handled by Vortex/Texis, not OpenSSL, and is only performed by clients (Vortex), not servers (Texis Monitor web server), as generally only servers have fixed, well-known hostnames and CNs that match.

A.2 Tokens and Messages

If any of the verification checks fail, an error message is issued with the details, in the form “Cannot verify certificate from host:port: reason at depth N”. The connection also fails with an SSL error for the peer (perhaps containing “alert bad certificate” or “alert unknown ca”). The depth N indicates how far from the peer leaf certificate the error occurred: depth 0 is the leaf certificate itself, depth 1 its issuer, etc.

The specific reason for the verification error can vary, e.g. certificate has expired, unable to get issuer certificate, etc. Any of these errors can be individually suppressed or enabled, by adding the string token for the error to the sslverifyserver or SSL Verify Client setting value, space-separated, prepended with a “+” (plus) or “-” (minus) sign. For example, to verify certificates but ignore expired and revoked certificate errors, the setting value would be “on -X509_V_ERR_CERT_HAS_EXPIRED -X509_V_ERR_CERT_REVOKED”. Note that ignoring any of the verification checks in this way may reduce security, and is generally only used in testing or debug environments.

The following list contains all the possible sslverifyserver or SSL Verify Client tokens that can be suppressed/enabled, and their corresponding error messages. Tokens starting with “X509_V_ERR...” correspond to the same-named OpenSSL certificate verification error:

- X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT
  unable to get issuer certificate
  A trusted certificate’s issuer certificate could not be found amongst the trusted certificates. This error usually means some trusted certificates’ chain(s) are incompletely set in sslcacertificatefile or SSL CA Certificate File, and this was discovered when trying to use these incompletely-chained certificate(s) to complete the peer chain. (See also X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT_LOCALLY.)

- X509_V_ERR_UNABLE_TO_GET_CRL
unable to get certificate CRL

- X509_V_ERR_UNABLE_TO_DECRIPT_CERT_SIGNATURE
unable to decrypt certificate’s signature

- X509_V_ERR_UNABLE_TO_DECRIPT_CRL_SIGNATURE
unable to decrypt CRL’s signature

- X509_V_ERR_UNABLE_TO_DECODE_ISSUER_PUBLIC_KEY
unable to decode issuer public key

- X509_V_ERR_CERT_SIGNATURE_FAILURE
certificate signature failure

- X509_V_ERR_CRL_SIGNATURE_FAILURE
CRL signature failure

- X509_V_ERR_CERT_NOT_YET_VALID
certificate is not yet valid

- X509_V_ERR_CERT_HAS_EXPIRED
certificate has expired

- X509_V_ERR_CRL_NOT_YET_VALID
CRL is not yet valid

- X509_V_ERR_CRL_HAS_EXPIRED
CRL has expired

- X509_V_ERR_ERROR_IN_CERT_NOT BEFORE_FIELD
format error in certificate’s notBefore field

- X509_V_ERR_ERROR_IN_CERT_NOT AFTER FIELD
format error in certificate’s notAfter field

- X509_V_ERR_ERROR_IN_CRL_LAST UPDATE_FIELD
format error in CRL’s lastUpdate field

- X509_V_ERR_ERROR_IN_CRL_NEXT UPDATE FIELD
format error in CRL’s nextUpdate field

- X509_V_ERR_OUT OF MEM
out of memory

- X509_V_ERR_DEPTH_ZERO_SELF_SIGNED_CERT
self signed certificate
Indicates that the peer certificate is self-signed, and is not trusted locally, i.e. is not in the local sslcacertificatefile or SSL CA Certificate File trust list.

- X509_V_ERR_SELF_SIGNED_CERT_IN_CHAIN
self signed certificate in certificate chain
Indicates that the peer chain’s root certificate is not trusted locally, i.e. is not in the local sslcacertificatefile or SSL CA Certificate File trust list.
A.2. TOKENS AND MESSAGES

- **X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT_LOCALLY**
  unable to get local issuer certificate
  A peer certificate’s issuer certificate was not found, after looking in both the chain provided by the peer and in the local trusted storage (i.e. `sslcacertificatefile` or `SSL CA Certificate File`). This usually means the peer chain was missing or incomplete.

- **X509_V_ERR_UNABLE_TO_VERIFY_LEAF_SIGNATURE**
  unable to verify the first certificate

- **X509_V_ERR_CERT_CHAIN_TOO_LONG**
  certificate chain too long
  The peer certificate’s chain length exceeded the limit set locally by `sslverifydepth` or `SSL Verify Depth`. The chain length (depth) is the number of certificates beyond the leaf (client or server) certificate itself; e.g. a chain of length 1 is a leaf certificate plus its issuer certificate.

- **X509_V_ERR_CERT_REVOKED**
  certificate revoked

- **X509_V_ERR_INVALID_CA**
  invalid CA certificate

- **X509_V_ERR_PATH_LENGTH_EXCEEDED**
  path length constraint exceeded

- **X509_V_ERR_INVALID_PURPOSE**
  unsupported certificate purpose

- **X509_V_ERR_CERT_UNTRUSTED**
  certificate not trusted

- **X509_V_ERR_CERT_REJECTED**
  certificate rejected

- **X509_V_ERR_SUBJECT_ISSUER_MISMATCH**
  subject issuer mismatch

- **X509_V_ERR_AKID_SKID_MISMATCH**
  authority and subject key identifier mismatch

- **X509_V_ERR_AKID_ISSUER_SERIAL_MISMATCH**
  authority and issuer serial number mismatch

- **X509_V_ERR_KEYUSAGE_NO_CERTSIGN**
  key usage does not include certificate signing

- **X509_V_ERR_UNABLE_TO_GET_CRL_ISSUER**
  unable to get CRL issuer certificate

- **X509_V_ERR_UNHANDLED_CRITICAL_EXTENSION**
  unhandled critical extension
• X509_V_ERR_KEYUSAGE_NO_CRL_SIGN  
  key usage does not include CRL signing

• X509_V_ERR_UNHANDLED_CRITICAL_CRL_EXTENSION  
  unhandled critical CRL extension

• X509_V_ERR_INVALID_NON_CA  
  invalid non-CA certificate (has CA markings)

• X509_V_ERR_PROXY_PATH_LENGTH_EXCEEDED  
  proxy path length constraint exceeded

• X509_V_ERR_KEYUSAGE_NO_DIGITAL_SIGNATURE  
  key usage does not include digital signature

• X509_V_ERR_PROXY_CERTIFICATES_NOT_ALLOWED  
  proxy certificates not allowed, please set the appropriate flag

• X509_V_ERR_INVALID_EXTENSION  
  invalid or inconsistent certificate extension

• X509_V_ERR_INVALID_POLICY_EXTENSION  
  invalid or inconsistent certificate policy extension

• X509_V_ERR_NO_EXPLICIT_POLICY  
  no explicit policy

• No_Peer_Certificate  
  Peer did not return a certificate
  The peer did not return a certificate when requested. Usually only occurs as an error on servers, indicating lack of a client certificate. (SSL requires server certificates\(^3\), whereas client certificates are optional. Thus clients would not see this error, as lack of a server certificate would mean the server would not have started in the first place.)

• CommonName_Hostname_Mismatch  
  Certificate Common Name ‘...’ does not match hostname ‘...’
  The peer certificate’s Common Name (CN) value did not match the hostname in the URL used to access it. This may indicate the host actually contacted is not the one expected by the URL, a possible security issue. This error is not possible for servers (nor possible to disable in SSL Verify Client), as servers do not do Common-Name-to-hostname matching: client certificates’ Common Names are not generally hostnames.

• Other_Error  
  Other/unknown X509_V_ERR_... error.

\(^3\)Unless an anonymous cipher is used, which is rarely if ever the case.
Appendix B

Trace Vortex

Trace Vortex allows Vortex developers to analyze Vortex execution. It includes timing information for every line, and the visualizing tool provides cumulative information for loops, functions. It can be very useful for determining which sections of a Vortex program are consuming time.

It consists of two parts - the logger, and the visualizer.

B.1 TraceVortex Logging

TraceVortex logging is controlled via the `tracevortex` and `tracevortex log vxcp` settings.

- `tracevortex log $file` - Sets the filename to use when TraceVortex logging is enabled. It defaults to the script path (sans extension) with a `.vstrace` extension in the script directory. Setting it to empty restores the default. Returns the previous value.

- `tracevortex on|off` - Starts or stops TraceVortex logging. Overwrites any existing log file. Logging also automatically finishes when the Vortex program exits.

The `.vstrace` log file is a list of statements executed, including timestamps, line numbers, statement information, and other data about the program’s execution. It isn’t meant to be read directly by humans, but instead can be opened by the visualization program to provide a more usable view of the data.

B.2 TraceVortex Visualizer

The TraceVortex visualizer is `TraceVortex.exe` in your installation directory. It also includes `TraceVortex.py`, the Python source of the program. Running the source directly requires `WxPython`, and `PyWin32` if you’re on Windows.

The TraceVortex Visualizer reads `.vstrace` log files produced by Vortex programs that enable `tracevortex` logging. It provides an interactive representation of the program’s execution, allowing you to see which parts of the program took various amounts of time.
.vstrace log files contain line number references to the Vortex source that ran. You’ll need the Vortex source code that made the log available when analyzing the .vstrace log.

**B.2.1 Overview**

After opening your .vstrace file, you’ll be presented with three main frames in the TraceVortex window, detailed below:

```
+------+-----------------+
| | |
| 1 | |
+--+| 3 |
| | |
| 2 | |
+--+-----------------+
```

- 1 - Function Calls Panel
- 2 - Functions by Caller Panel
- 3 - Tree View Panel

**B.2.2 Function Calls Panel**

The Function Calls panel lists information on all of the function calls performed in during the logging. It consists of 3 columns:

- **Function** - The name of the function that was invoked. This can be either a script-defined function (such as `readFormVars`), or a built-in function, which will appear with angle brackets (such as `<SQL>`).
- **Calls** - The number of times this function was called throughout the execution.
- **Duration** - The total duration of all calls made to this function. A function that calls itself is not doubly-counted; Duration is the amount of time the program spent with this function somewhere in its call stack.

Double-clicking on any function will list all of the locations that called this function in the “Functions by Caller” panel below.
B.2. TRACEVORTEX VISUALIZER

B.2.3 Functions by Caller Panel

Functions by Caller gives information about every time a given function was invoked. You can look up a function by typing the name and pressing Find Callers, or by double-clicking on a function in the Function Calls panel above.

For every invocation, the panel lists:

- **Timestamp** - The timestamp of when this call occurred. This is recorded from the start of the Vortex process, not the start of logging. If logging is started after a program has been running 18 seconds, timestamps will start with a value like 18.047970.

- **Duration** - The duration of this statement. If this is a function call or a looping statement, it includes the time of all operations inside the function, until the point that the function returns.

- **Called By** - The parent function that performed the invocation of this function.

Double-clicking on an invocation will expand the Tree View Panel to the point where that call occurred.

B.2.4 Tree View Panel

After opening your .vstrace file, you’ll be presented with a tree view of program execution as a series of statements, some containing others. The lines contain multiple fields, which are:

```
1.047970 490ms (1.2%) process.vs:26 <readFormVars>
```

- **1** - The timestamp of when this statement occurred. This is recorded from the start of the Vortex process, not the start of logging. If logging is started after a program has been running 18 seconds, timestamps will start with a value like 18.047970.

- **2** - The duration of this statement. If this is a function call or a looping statement, it includes the time of all operations inside the function, until the point that the function returns or the loop finishes.

- **3** - The percentage of the parent’s total time that this statement constitutes. If `<processPage>` takes 3ms and calls `<readPage>`, which takes 2ms, the latter will show as 66%.

  This allows users to quickly see which statements in a function are more responsible for the collective time of the whole function call.

- **4** - The Vortex source file and line number this statement came from. A single .vstrace log may span multiple source files if one script `<use>`s others.

- **5** - The Vortex source at that line. Not all executed statements exist in the source, such as implicit `<return>` statements at the end of functions.

B.2.5 Preferences

Preferences can be changed by going to File -> Preferences... in the menu, or by clicking the Preferences button on the toolbar. Changing any preferences will automatically reload the .vstrace.
APPENDIX B. TRACE VORTEX

• Discard <local> levels (on)

The vortex <local> statement is used to define local variables. Many functions begin with a local declaration, which can result in lots of extraneous expanding - find a function, expand local, find next function, expand local, etc, even though these <local> statements don’t add much to the run time.

If Discard <local> statements is on, these <local> statements are ignored by the visualizer, giving a much more concise view of the execution.

• Merge calls/loops faster than ___ µs (off)

Vortex execution often has many statements that execute very quickly and a small number that take much longer, and it’s the latter that we’re interested in. Keeping track of all of the tiny, inconsequential statements can bloat RAM usage of the visualizer.

If Merge calls/loops is checked, details about any functions or loops that execute faster than 500 microseconds (µs) is discarded. The call itself is still present, but the operations of that function will not be expandable.

Statements that have been merged will end with text like (23 Merged), letting you know how many statements beneath it were discarded.

You can customize the threshold in the text box; the higher it is, the more information will be discarded, and the less RAM will be used by the visualizer.

Here’s an example with the default threshold enabled. <processPage> took 660 µs, so we’ll be able to expand it to see what it called. It called <readPage>, which took 116 µs. We won’t be able to expand <readPage> to see exactly what it called, because it fell beneath our threshold and got merged. If the threshold is decreased to 100 µs or turned off, we could expand <readPage>. If the threshold is increased to 900 µs, then <processPage> would have been merged, and we wouldn’t have been able to expand it to see that it called <readPage>.

• Group adjacent statements (off)

Function calls and loops are often what we’re look for in the visualizer, which can lead to scrolling past many single statements, like assigning or printing.

Setting Group adjacent statements on will cause all sequential non-looping, non-function call statements to display as a single collapsed “group”, which can then be expanded.

Grouping differs from the merging setting above in that it never “loses” statements. Grouping will never affect loops or function calls. Grouping will only cause a long list of single statements to be displayed as an expandable group.

Grouping statements can provide more compact browsing, and uses slightly less RAM.

• Show Vortex statements in Function Calls panel (off)

By default, the list of functions in the Function Calls panel in the upper left only contains functions defined in the script. Enabling this causes TraceVortex to also show native vortex functions, like <SQL> and <REX>, in the list of functions.

This can clutter the function list, but it can also help shed light on which vortex statements are consuming time, rather than just which script functions.
• **Show .vstrace line numbers** *(off)*
  
  This causes statements to also show the line number that they occur in the raw .vstrace file. This normally isn’t needed.

• **Date Source** *(real)*
  
  This determines what TraceVortex uses for timing / information.
  
  – **user** - The amount of time spent executing “user” code (from the OS perspective).
  
  – **sys** - The amount of time spent in kernel, waiting on system calls (like I/O).
  
  – **real** - The amount of wall-clock time elapsed. This can be affected by waiting for other processes or actions on the machine, and represents the actual time taken.
  
  – **user+sys** - The total of user and sys added together.
Appendix C

Third-Party Software

Webinator may contain and utilize the following third-party software to enhance its functionality, depending on the version purchased. Note that your usage and rights to such third-party software may be governed by the appropriate licenses originating with that software, in addition to your License Agreement with Thunderstone - EPI for Thunderstone software.

C.1 Antiword

The antiword package is used by Thunderstone’s anytotx plugin to handle Microsoft(R) Word files. It has been modified to work within anytotx's installation and to extract meta information. Thunderstone’s modified source may be obtained from ftp://ftp.thunderstone.com/pub/epi-gpl/msfilt.tar.gz or by contacting Thunderstone tech support and requesting a CD containing the modified Antiword source. Sending a CD will require payment of shipping and handling charges by the requestor. antiword is governed by the terms of the GNU GPL, which is reproduced on p. 625.

C.2 Aspell

The GNU Project's aspell package is executed by (but not linked or compiled into) Webinator for spell-checking and “Did you mean...” queries. Complete source code and documentation is available at ftp://ftp.thunderstone.com/pub/epi-gpl/aspell-0.50.3.tar.gz or ftp://ftp.thunderstone.com/pub/epi-gpl/aspell-0.60.4.tar.gz or by contacting Thunderstone tech support and requesting a CD containing the source. Sending of a CD will require payment of shipping and handling charges by the requestor. aspell is governed by the terms of the GNU Lesser GPL, which is reproduced on p. 642.
C.3 Catdoc xls2csv

Catdoc’s xls2csv program is used by Thunderstone’s anytotx plugin to handle Microsoft(R) Excel(R) spreadsheet files. It has been modified to work within anytotx’s installation and to extract meta information. Thunderstone’s modified source may be obtained from ftp://ftp.thunderstone.com/pub/epi-gpl/msfilt.tar.gz or by contacting Thunderstone tech support and requesting a CD containing the modified Catdoc source. Sending a CD will require payment of shipping and handling charges by the requestor. Catdoc is governed by the terms of the GNU GPL, which is reproduced on p. 625.

C.4 Cole library

The cole library is used by Thunderstone’s versions of catdoc and antiword. It has been modified to prevent extraneous printing. Thunderstone’s modified source may be obtained from ftp://ftp.thunderstone.com/pub/epi-gpl/msfilt.tar.gz or by contacting Thunderstone tech support and requesting a CD containing the modified cole source. Sending a CD will require payment of shipping and handling charges by the requestor. The cole library is governed by the terms of the GNU GPL, which is reproduced on p. 625.

C.5 iconv

GNU libiconv may be used by Thunderstone’s HTML processor to convert documents in certain character sets. GNU libiconv is not incorporated into Thunderstone’s products but is a separate standalone program, called via exec() and writing/reading standard input/output. You may obtain complete source code and documentation for libiconv at ftp://ftp.thunderstone.com/pub/epi-gpl/libiconv-1.9.2.tar.gz or by contacting Thunderstone tech support and requesting a CD containing the GNU libiconv source. Sending a CD will require payment of shipping and handling charges by the requestor. GNU libiconv is governed by the terms of the GNU Library GPL, which is reproduced on p. 642.

C.6 libpst

The readpst program included in the libpst package may be used by Thunderstone’s anytotx program to convert PST (Personal Storage Table) files from Microsoft Outlook. Complete source code for libpst may be obtained at ftp://ftp.thunderstone.com/pub/epi-gpl/libpst-0.6.55.tar.gz or by contacting Thunderstone tech support and requesting a CD containing the libpst source. Sending a CD will require payment of shipping and handling charges by the requestor. libpst is governed by the terms of the GNU GPL version 2, which is reproduced on p. 625.
C.7  libxml2

Libxml2 may be used by Thunderstone’s products to parse XML documents. It is available at http://xmlsoft.org/, and is Copyright (C) 1998-2003 Daniel Veillard. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of libxml2 and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE DANIEL VEILLARD BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Except as contained in this notice, the name of Daniel Veillard shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization from him.

C.8  Libxslt

Libxslt may be used by Thunderstone’s products to apply XSL transforms to XML documents. It is available at http://xmlsoft.org/XSLT/ and is Copyright (C) 2001-2002 Daniel Veillard. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of libxslt and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE DANIEL VEILLARD BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Except as contained in this notice, the name of Daniel Veillard shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization from him.
APPENDIX C. THIRD-PARTY SOFTWARE

C.9 Libexslt

Libexslt may be used by Thunderstone’s products when applying XSL transforms to XML documents. It is Copyright (C) 2001-2002 Thomas Broyer, Charlie Bozeman and Daniel Veillard. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of libexslt and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Except as contained in this notice, the name of the authors shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization from them.

C.10 ppt2html, msg2html

ppt2html and msg2html may be used by Thunderstone’s anytotx document filter to convert Microsoft(R) PowerPoint and .msg files. Source is available at:


or by contacting Thunderstone tech support and requesting a CD containing the source. Sending a CD will require payment of shipping and handling charges by the requestor. ppt2html and msg2html are governed by the terms of the GNU GPL, which is reproduced on p. 625.

C.11 SSL/HTTPS plugin

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/). Copyright ©1998-2011 The OpenSSL Project. All rights reserved. This product includes cryptographic software written by Eric Young (eay@cryptsoft.com). Copyright ©1995-1998 Eric Young. All rights reserved.

The OpenSSL toolkit stays under a dual license, i.e. both the
conditions of the OpenSSL License and the original SSLeay license
apply to the toolkit. See below for the actual license
texts. Actually both licenses are BSD-style Open Source licenses. In
case of any license issues related to OpenSSL please contact
openssl-core@openssl.org.

OpenSSL License
--------------

Copyright (c) 1998-2011 The OpenSSL Project. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in
   the documentation and/or other materials provided with the
   distribution.

3. All advertising materials mentioning features or use of this
   software must display the following acknowledgment:
   "This product includes software developed by the OpenSSL Project
   for use in the OpenSSL Toolkit. (http://www.openssl.org/)"

4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
   endorse or promote products derived from this software without
   prior written permission. For written permission, please contact
   openssl-core@openssl.org.

5. Products derived from this software may not be called "OpenSSL"
   nor may "OpenSSL" appear in their names without prior written
   permission of the OpenSSL Project.

6. Redistributions of any form whatsoever must retain the following
   acknowledgment:
   "This product includes software developed by the OpenSSL Project
   for use in the OpenSSL Toolkit (http://www.openssl.org/)"

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT "AS IS" AND ANY
EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR
ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
OF THE POSSIBILITY OF SUCH DAMAGE.

This product includes cryptographic software written by Eric Young
(eay@cryptsoft.com). This product includes software written by Tim
Hudson (tjh@cryptsoft.com).

Original SSLeay License
--------------------

Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
All rights reserved.

This package is an SSL implementation written
by Eric Young (eay@cryptsoft.com).
The implementation was written so as to conform with Netscapes SSL.

This library is free for commercial and non-commercial use as long as
the following conditions are heared to. The following conditions
apply to all code found in this distribution, be it the RC4, RSA,
lhash, DES, etc., code; not just the SSL code. The SSL documentation
included with this distribution is covered by the same copyright terms
except that the holder is Tim Hudson (tjh@cryptsoft.com).

Copyright remains Eric Young’s, and as such any Copyright notices in
the code are not to be removed. If this package is used in a product,
Eric Young should be given attribution as the author of the parts of
the library used. This can be in the form of a textual message at
program startup or in documentation (online or textual) provided with
the package.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software
   must display the following acknowledgement:
   "This product includes cryptographic software written by
   Eric Young (eay@cryptsoft.com)"
The word ‘cryptographic’ can be left out if the rouines from the
library being used are not cryptographic related :-).
4. If you include any Windows specific code (or a derivative thereof)
from the apps directory (application code) you must include an
acknowledgement: "This product includes software written by
Tim Hudson (tjh@cryptsoft.com)"
C.12. UNRAR

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The licence and distribution terms for any publically available version or derivative of this code cannot be changed. i.e. this code cannot simply be copied and put under another distribution licence [including the GNU Public Licence.]

C.12 unrar

The Thunderstone file converter plugin (anytotx) may utilize Alexander L. Roshal’s unrar utility to unpack RAR archive files (*.rar). The unrar utility is governed by the unRAR license reproduced below:

****** ***** ****** unRAR - free utility for RAR archives
** ** ** ** ** ** ------------------------------
****** ***** ****** License for use and distribution of
** ** ** ** ** ** ------------------------------
** ** ** ** ** ** FREE portable version
---------------------

The source code of unRAR utility is freeware. This means:

1. All copyrights to RAR and the utility unRAR are exclusively owned by the author - Alexander Roshal.

2. The unRAR sources may be used in any software to handle RAR archives without limitations free of charge, but cannot be used to re-create the RAR compression algorithm, which is proprietary. Distribution of modified unRAR sources in separate form or as a part of other software is permitted, provided that it is clearly stated in the documentation and source comments that the code may not be used to develop a RAR (WinRAR) compatible archiver.

3. The unRAR utility may be freely distributed. No person or company may charge a fee for the distribution of unRAR without written permission from the copyright holder.

4. THE RAR ARCHIVER AND THE UNRAR UTILITY ARE DISTRIBUTED "AS IS". NO WARRANTY OF ANY KIND IS EXPRESSED OR IMPLIED. YOU USE AT YOUR OWN RISK. THE AUTHOR WILL NOT BE LIABLE FOR DATA LOSS,
DAMAGES, LOSS OF PROFITS OR ANY OTHER KIND OF LOSS WHILE USING OR MISUSING THIS SOFTWARE.

5. Installing and using the unRAR utility signifies acceptance of these terms and conditions of the license.

6. If you don’t agree with terms of the license you must remove unRAR files from your storage devices and cease to use the utility.

Thank you for your interest in RAR and unRAR.

Alexander L. Roshal

C.13 unzip

The Thunderstone file converter plugin (anytotx) may utilize Info-ZIP’s unzip utility to unpack ZIP archive files (*.zip). The unzip software is governed by the Info-ZIP license reproduced below:

This is version 2002-Feb-16 of the Info-ZIP copyright and license. The definitive version of this document should be available at ftp://ftp.info-zip.org/pub/infozip/license.html indefinitely.

Copyright (c) 1990-2002 Info-ZIP. All rights reserved.

For the purposes of this copyright and license, "Info-ZIP" is defined as the following set of individuals:


This software is provided "as is," without warranty of any kind, express or implied. In no event shall Info-ZIP or its contributors be held liable for any direct, indirect, incidental, special or consequential damages arising out of the use of or inability to use this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. Redistributions of source code must retain the above copyright notice, definition, disclaimer, and this list of conditions.

2. Redistributions in binary form (compiled executables) must reproduce
the above copyright notice, definition, disclaimer, and this list of conditions in documentation and/or other materials provided with the distribution. The sole exception to this condition is redistribution of a standard UnZipSFX binary as part of a self-extracting archive; that is permitted without inclusion of this license, as long as the normal UnZipSFX banner has not been removed from the binary or disabled.

3. Altered versions—including, but not limited to, ports to new operating systems, existing ports with new graphical interfaces, and dynamic, shared, or static library versions—must be plainly marked as such and must not be misrepresented as being the original source. Such altered versions also must not be misrepresented as being Info-ZIP releases—including, but not limited to, labeling of the altered versions with the names "Info-ZIP" (or any variation thereof, including, but not limited to, different capitalizations), "Pocket UnZip," "WiZ" or "MacZip" without the explicit permission of Info-ZIP. Such altered versions are further prohibited from misrepresentative use of the Zip-Bugs or Info-ZIP e-mail addresses or of the Info-ZIP URL(s).


C.14 zlib

Webinator utilizes the zlib compression library, with the following license:

Copyright (C) 1995-2003 Jean-loup Gailly and Mark Adler.

This software is provided ‘as-is’, without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Jean-loup Gailly         Mark Adler  
jloup@gzip.org         madler@alumni.caltech.edu
The zlib library data format is described by RFCs (Request for Comments) 1950 to 1952 in the files http://www.ietf.org/rfc/rfc1950.txt (zlib format), rfc1951.txt (deflate format) and rfc1952.txt (gzip format).

C.15 SpiderMonkey (JavaScript-C) Engine

The libtxjs.* library (Thunderstone JavaScript plugin) contains and utilizes the SpiderMonkey engine, as well as additional functionality.

The txjs.tar file contains context diffs (patches) to the Mozilla Project’s SpiderMonkey (JavaScript-C) engine, version 1.5-rc4. Complete documentation and source code to the SpiderMonkey Engine is available at http://www.mozilla.org/js/spidermonkey/.

The patches in txjs.tar were created by Thunderstone Software LLC and apply to the core SpiderMonkey engine. They are provided for compliance with the Netscape Public License, which governs usage of the SpiderMonkey engine. A copy of the Netscape Public License is on p. 651. Note that the libtxjs.* library also contains other (Thunderstone) code.

C.16 PDF/anytotx plugin

Portions of this product Copyright 1996-2000 Glyph & Cog, LLC.

C.17 thttpd - throttling HTTP server

Webinator’s vhttpd web server is derived in part from thttpd, Copyright ©1995 by Jef Poskanzer jef@acme.com. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ‘‘AS IS’’ AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

C.18 prngd


C.19 GNU General Public License

Some third-party software packages shipped with Webinator are governed by the GNU General Public License, reproduced below. See the Third-Party Software section, p. 615, for a list of applicable packages.

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
675 Mass Ave, Cambridge, MA 02139, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation’s software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.
For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author’s protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors’ reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone’s free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program’s source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License
along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
APPENDIX C. THIRD-PARTY SOFTWARE

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.
6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to
address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively
convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program’s name and a brief idea of what it does.>
Copyright (C) 19yy  <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

<table>
<thead>
<tr>
<th>Gnomovision version 69, Copyright (C) 19yy name of author</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.</td>
</tr>
<tr>
<td>This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.</td>
</tr>
</tbody>
</table>

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than 'show w' and 'show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

<table>
<thead>
<tr>
<th>Yoyodyne, Inc., hereby disclaims all copyright interest in the program 'Gnomovision' (which makes passes at compilers) written by James Hacker.</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;signature of Ty Coon&gt;, 1 April 1989</td>
</tr>
<tr>
<td>Ty Coon, President of Vice</td>
</tr>
</tbody>
</table>

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use
the GNU Library General Public License instead of this License.

C.20 GNU Lesser General Public License

Some third-party software packages distributed with Webinator are governed by the GNU Lesser General Public License, reproduced below. See the Third-Party Software section, p. 615, for a list of applicable packages.

GNU LESSER GENERAL PUBLIC LICENSE
Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source
code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free
software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users’ freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for
writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library’s complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a) The modified work must itself be a software library.

   b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

   c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

   d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

   (For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote
Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.
When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if
the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

   a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

   b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses
terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients’ exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the
original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS
How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the library’s name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc.,
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the library ‘Frob’ (a library for tweaking knobs) written by James Random Hacker.

<signature of Ty Coon>, 1 April 1990
Ty Coon, President of Vice

That’s all there is to it!
C.21 GNU Library General Public License

Some third-party software packages distributed with Webinator are governed by the GNU Library General Public License, reproduced below. See the Third-Party Software section, p. 615, for a list of applicable packages.

GNU LIBRARY GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the library GPL. It is numbered 2 because it goes with version 2 of the ordinary GPL.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Library General Public License, applies to some specially designated Free Software Foundation software, and to any other libraries whose authors decide to use it. You can use it for your libraries, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library, or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link a program with the library, you must provide complete object files to the recipients so that they can relink them with the library, after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

Our method of protecting your rights has two steps: (1) copyright the library, and (2) offer you this license which gives you legal
permission to copy, distribute and/or modify the library.

Also, for each distributor’s protection, we want to make certain that everyone understands that there is no warranty for this free library. If the library is modified by someone else and passed on, we want its recipients to know that what they have is not the original version, so that any problems introduced by others will not reflect on the original authors’ reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that companies distributing free software will individually obtain patent licenses, thus in effect transforming the program into proprietary software. To prevent this, we have made it clear that any patent must be licensed for everyone’s free use or not licensed at all.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License, which was designed for utility programs. This license, the GNU Library General Public License, applies to certain designated libraries. This license is quite different from the ordinary one; be sure to read it in full, and don’t assume that anything in it is the same as in the ordinary license.

The reason we have a separate public license for some libraries is that they blur the distinction we usually make between modifying or adding to a program and simply using it. Linking a program with a library, without changing the library, is in some sense simply using the library, and is analogous to running a utility program or application program. However, in a textual and legal sense, the linked executable is a combined work, a derivative of the original library, and the ordinary General Public License treats it as such.

Because of this blurred distinction, using the ordinary General Public License for libraries did not effectively promote software sharing, because most developers did not use the libraries. We concluded that weaker conditions might promote sharing better.

However, unrestricted linking of non-free programs would deprive the users of those programs of all benefit from the free status of the libraries themselves. This Library General Public License is intended to permit developers of non-free programs to use free libraries, while preserving your freedom as a user of such programs to change the free libraries that are incorporated in them. (We have not seen how to achieve this as regards changes in header files, but we have achieved it as regards changes in the actual functions of the Library.) The hope is that this will lead to faster development of free libraries.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, while the latter only works together with the library.
Note that it is possible for a library to be covered by the ordinary General Public License rather than by this special one.

GNU LIBRARY GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Library General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library’s complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion
of it, thus forming a work based on the Library, and copy and
distribute such modifications or work under the terms of Section 1
above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices
stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no
charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a
table of data to be supplied by an application program that uses
the facility, other than as an argument passed when the facility
is invoked, then you must make a good faith effort to ensure that,
in the event an application does not supply such function or
table, the facility still operates, and performs whatever part of
its purpose remains meaningful.

(For example, a function in a library to compute square roots has
a purpose that is entirely well-defined independent of the
application. Therefore, Subsection 2d requires that any
application-supplied function or table used by this function must
be optional: if the application does not supply it, the square
root function must still compute square roots.)

These requirements apply to the modified work as a whole. If
identifiable sections of that work are not derived from the Library,
and can be reasonably considered independent and separate works in
themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works. But when you
distribute the same sections as part of a whole which is a work based
on the Library, the distribution of the whole must be on the terms of
this License, whose permissions for other licensees extend to the
entire whole, and thus to each and every part regardless of who wrote
it.

Thus, it is not the intent of this section to claim rights or contest
your rights to work written entirely by you; rather, the intent is to
exercise the right to control the distribution of derivative or
collective works based on the Library.

In addition, mere aggregation of another work not based on the Library
with the Library (or with a work based on the Library) on a volume of
a storage or distribution medium does not bring the other work under
the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public
License instead of this License to a given copy of the Library. To do
this, you must alter all the notices that refer to this License, so
that they refer to the ordinary GNU General Public License, version 2,
Instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may
distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also compile or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

   a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

   b) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

   c) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

   d) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.
It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients’ exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot
distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Library General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our
decision will be guided by the two goals of preserving the free status
of all derivatives of our free software and of promoting the sharing
and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO
WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW.
EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR
OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY
KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE
LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME
THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN
WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY
AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU
FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR
CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE
LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING
RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A
FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF
SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH
DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest
possible use to the public, we recommend making it free software that
everyone can redistribute and change. You can do so by permitting
redistribution under these terms (or, alternatively, under the terms
of the ordinary General Public License).

To apply these terms, attach the following notices to the library.
It is safest to attach them to the start of each source file to most
effectively convey the exclusion of warranty; and each file should
have at least the "copyright" line and a pointer to where the full
notice is found.

<one line to give the library’s name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Library General Public
License as published by the Free Software Foundation; either
version 2 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
Library General Public License for more details.

You should have received a copy of the GNU Library General Public
License along with this library; if not, write to the Free
Software Foundation, Inc., 59 Temple Place - Suite 330, Boston,
MA 02111-1307, USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the library, if
necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the library
"Frob" (a library for tweaking knobs) written by James Random Hacker.

<signature of Ty Coon>, 1 April 1990
Ty Coon, President of Vice

That’s all there is to it!

C.22 Netscape Public License

Some third-party software packages distributed with Webinator are governed by the Netscape Public
License, reproduced below. See the Third-Party Software section, p. 615, for a list of applicable packages.

Netscape Public License version 1.1

AMENDMENTS The Netscape Public License Version 1.1 ("NPL") consists of the Mozilla Public
License Version 1.1 with the following Amendments, including Exhibit A-Netscape Public License.
Files identified with "Exhibit A-Netscape Public License" are governed by the Netscape Public
License Version 1.1.

Additional Terms applicable to the Netscape Public License.

I. Effect.

These additional terms described in this Netscape Public License – Amendments shall apply to the Mozilla
Communicator client code and to all Covered Code under this License.

II. "Netscape’s Branded Code" means Covered Code that Netscape distributes and/or permits others to
distribute under one or more trademark(s) which are controlled by Netscape but which are not licensed for
use under this License.

III. Netscape and logo. This License does not grant any rights to use the trademarks "Netscape", the
"Netscape Navigator" logo or the "Netcenter" logo, "Gecko", "Java" or
"JavaScript", "Smart Browsing" even if such marks are included in the Original Code or Modifications.

IV. Inability to Comply Due to Contractual Obligation. Prior to licensing the Original Code under this
License, Netscape has licensed third party code for use in Netscape’s Branded Code. To the extent that Netscape is limited contractually from making such third party code available under this License, Netscape may choose to reintegrate such code into Covered Code without being required to distribute such code in Source Code form, even if such code would otherwise be considered “Modifications” under this License.

V. Use of Modifications and Covered Code by Initial Developer.

V.1. In General. The obligations of Section 3 apply to Netscape, except to the extent specified in this Amendment, Section V.2 and V.3.

V.2. Other Products. Netscape may include Covered Code in products other than the Netscape’s Branded Code which are released by Netscape during the two (2) years following the release date of the Original Code, without such additional products becoming subject to the terms of this License, and may license such additional products on different terms from those contained in this License.

V.3. Alternative Licensing. Netscape may license the Source Code of Netscape’s Branded Code, including Modifications incorporated therein, without such Netscape Branded Code becoming subject to the terms of this License, and may license such Netscape Branded Code on different terms from those contained in this License.

VI. Litigation. Notwithstanding the limitations of Section 11 above, the provisions regarding litigation in Section 11(a), (b) and (c) of the License shall apply to all disputes relating to this License.

EXHIBIT A-Netscape Public License.

"The contents of this file are subject to the Netscape Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.mozilla.org/NPL/ Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is Mozilla Communicator client code, released March 31, 1998.

The Initial Developer of the Original Code is Netscape Communications Corporation. Portions created by Netscape are Copyright (C) 1998-1999 Netscape Communications Corporation. All Rights Reserved. Contributor(s): ————————————–.

Alternatively, the contents of this file may be used under the terms of the —— license (the "[—] License"), in which case the provisions of [——] License are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of the [——] License and not to allow others to use your version of this file under the NPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the [—] License. If you do not delete the provisions above, a recipient may use your version of this file under either the NPL or the [—] License.”

MOZILLA PUBLIC LICENSE

Version 1.1

1. Definitions.

1.0.1. ”Commercial Use” means distribution or otherwise making the Covered Code available to a third party.
1.1. "Contributor" means each entity that creates or contributes to the creation of Modifications.

1.2. "Contributor Version" means the combination of the Original Code, prior Modifications used by a Contributor, and the Modifications made by that particular Contributor.

1.3. "Covered Code" means the Original Code or Modifications or the combination of the Original Code and Modifications, in each case including portions thereof.

1.4. "Electronic Distribution Mechanism" means a mechanism generally accepted in the software development community for the electronic transfer of data.

1.5. "Executable" means Covered Code in any form other than Source Code.

1.6. "Initial Developer" means the individual or entity identified as the Initial Developer in the Source Code notice required by Exhibit A.

1.8. "License" means this document.

1.8.1. "Licensable" means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently acquired, any and all of the rights conveyed herein.

1.9. "Modifications" means any addition to or deletion from the substance or structure of either the Original Code or any previous Modifications. When Covered Code is released as a series of files, a Modification is:

A. Any addition to or deletion from the contents of a file containing Original Code or previous Modifications.

B. Any new file that contains any part of the Original Code or previous Modifications.

1.10. "Original Code" means Source Code of computer software code which is described in the Source Code notice required by Exhibit A as Original Code, and which, at the time of its release under this License is not already Covered Code governed by this License.

1.10.1. "Patent Claims" means any patent claim(s), now owned or hereafter acquired, including without limitation, method, process, and apparatus claims, in any patent Licensable by grantor.

1.11. "Source Code" means the preferred form of the Covered Code for making modifications to it, including all modules it contains, plus any associated interface definition files, scripts used to control compilation and installation of an Executable, or source code differential comparisons against either the Original Code or another well known, available Covered Code of the Contributor's choice. The Source Code can be in a compressed or archival form, provided the appropriate decompression or de-archiving software is widely available for no charge.

1.12. "You" (or "Your") means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License or a future version of this License issued under Section 6.1. For legal entities, "You" includes any entity which controls, is controlled by, or is under common control with You. For purposes of this definition, "control" means (a) the power, direct or indirect, to cause the direction or
management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) beneficial ownership of such entity.

2. Source Code License.

2.1. The Initial Developer Grant.

The Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license, subject to third party intellectual property claims:

(a) under intellectual property rights (other than patent or trademark) Licensable by Initial Developer to use, reproduce, modify, display, perform, sublicense and distribute the Original Code (or portions thereof) with or without Modifications, and/or as part of a Larger Work; and

(b) under Patents Claims infringed by the making, using or selling of Original Code, to make, have made, use, practice, sell, and offer for sale, and/or otherwise dispose of the Original Code (or portions thereof).

(c) the licenses granted in this Section 2.1(a) and (b) are effective on the date Initial Developer first distributes Original Code under the terms of this License.

(d) Notwithstanding Section 2.1(b) above, no patent license is granted: 1) for code that You delete from the Original Code; 2) separate from the Original Code; or 3) for infringements caused by: i) the modification of the Original Code or ii) the combination of the Original Code with other software or devices.

2.2. Contributor Grant.

Subject to third party intellectual property claims, each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license

(a) under intellectual property rights (other than patent or trademark) Licensable by Contributor, to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by such Contributor (or portions thereof) either on an unmodified basis, with other Modifications, as Covered Code and/or as part of a Larger Work; and

(b) under Patent Claims infringed by the making, using, or selling of Modifications made by that Contributor either alone and/or in combination with its Contributor Version (or portions of such combination), to make, use, sell, offer for sale, have made, and/or otherwise dispose of: 1) Modifications made by that Contributor (or portions thereof); and 2) the combination of Modifications made by that Contributor with its Contributor Version (or portions of such combination).

(c) the licenses granted in Sections 2.2(a) and 2.2(b) are effective on the date Contributor first makes Commercial Use of the Covered Code.

(d) Notwithstanding Section 2.2(b) above, no patent license is granted: 1) for any code that Contributor has deleted from the Contributor Version; 2) separate from the Contributor Version; 3) for infringements caused by: i) third party modifications of Contributor Version or ii) the combination of Modifications made by that Contributor with other software (except as part of the Contributor Version) or other devices; or 4) under Patent Claims infringed by Covered Code in the absence of Modifications made by that Contributor.

3. Distribution Obligations.

3.1. Application of License.

The Modifications which You create or to which You contribute are governed by the terms of this License,
including without limitation Section 2.2. The Source Code version of Covered Code may be distributed only under the terms of this License or a future version of this License released under Section 6.1, and You must include a copy of this License with every copy of the Source Code You distribute. You may not offer or impose any terms on any Source Code version that alters or restricts the applicable version of this License or the recipients’ rights hereunder. However, You may include an additional document offering the additional rights described in Section 3.5.

3.2. Availability of Source Code.

Any Modification which You create or to which You contribute must be made available in Source Code form under the terms of this License either on the same media as an Executable version or via an accepted Electronic Distribution Mechanism to anyone to whom you made an Executable version available; and if made available via Electronic Distribution Mechanism, must remain available for at least twelve (12) months after the date it initially became available, or at least six (6) months after a subsequent version of that particular Modification has been made available to such recipients. You are responsible for ensuring that the Source Code version remains available even if the Electronic Distribution Mechanism is maintained by a third party.

3.3. Description of Modifications.

You must cause all Covered Code to which You contribute to contain a file documenting the changes You made to create that Covered Code and the date of any change. You must include a prominent statement that the Modification is derived, directly or indirectly, from Original Code provided by the Initial Developer and including the name of the Initial Developer in (a) the Source Code, and (b) in any notice in an Executable version or related documentation in which You describe the origin or ownership of the Covered Code.

3.4. Intellectual Property Matters

(a) Third Party Claims.

If Contributor has knowledge that a license under a third party’s intellectual property rights is required to exercise the rights granted by such Contributor under Sections 2.1 or 2.2, Contributor must include a text file with the Source Code distribution titled “LEGAL” which describes the claim and the party making the claim in sufficient detail that a recipient will know whom to contact. If Contributor obtains such knowledge after the Modification is made available as described in Section 3.2, Contributor shall promptly modify the LEGAL file in all copies Contributor makes available thereafter and shall take other steps (such as notifying appropriate mailing lists or newsgroups) reasonably calculated to inform those who received the Covered Code that new knowledge has been obtained.

(b) Contributor APIs.

If Contributor’s Modifications include an application programming interface and Contributor has knowledge of patent licenses which are reasonably necessary to implement that API, Contributor must also include this information in the LEGAL file.

(c) Representations.

Contributor represents that, except as disclosed pursuant to Section 3.4(a) above, Contributor believes that Contributor’s Modifications are Contributor’s original creation(s) and/or Contributor has sufficient rights to grant the rights conveyed by this License.

3.5. Required Notices.
You must duplicate the notice in Exhibit A in each file of the Source Code. If it is not possible to put such notice in a particular Source Code file due to its structure, then You must include such notice in a location (such as a relevant directory) where a user would be likely to look for such a notice. If You created one or more Modification(s) You may add your name as a Contributor to the notice described in Exhibit A. You must also duplicate this License in any documentation for the Source Code where You describe recipients’ rights or ownership rights relating to Covered Code. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Code. However, You may do so only on Your own behalf, and not on behalf of the Initial Developer or any Contributor. You must make it absolutely clear than any such warranty, support, indemnity or liability obligation is offered by You alone, and You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of warranty, support, indemnity or liability terms You offer.

3.6. Distribution of Executable Versions.

You may distribute Covered Code in Executable form only if the requirements of Section 3.1-3.5 have been met for that Covered Code, and if You include a notice stating that the Source Code version of the Covered Code is available under the terms of this License, including a description of how and where You have fulfilled the obligations of Section 3.2. The notice must be conspicuously included in any notice in an Executable version, related documentation or collateral in which You describe recipients’ rights relating to the Covered Code. You may distribute the Executable version of Covered Code or ownership rights under a license of Your choice, which may contain terms different from this License, provided that You are in compliance with the terms of this License and that the license for the Executable version does not attempt to limit or alter the recipient’s rights in the Source Code version from the rights set forth in this License. If You distribute the Executable version under a different license You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by the Initial Developer or any Contributor. You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of any such terms You offer.

3.7. Larger Works.

You may create a Larger Work by combining Covered Code with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Code.

4. Inability to Comply Due to Statute or Regulation.

If it is impossible for You to comply with any of the terms of this License with respect to some or all of the Covered Code due to statute, judicial order, or regulation then You must: (a) comply with the terms of this License to the maximum extent possible; and (b) describe the limitations and the code they affect. Such description must be included in the LEGAL file described in Section 3.4 and must be included with all distributions of the Source Code. Except to the extent prohibited by statute or regulation, such description must be sufficiently detailed for a recipient of ordinary skill to be able to understand it.

5. Application of this License.

This License applies to code to which the Initial Developer has attached the notice in Exhibit A and to related Covered Code.

6. Versions of the License.
6.1. New Versions.

Netscape Communications Corporation ("Netscape") may publish revised and/or new versions of the License from time to time. Each version will be given a distinguishing version number.

6.2. Effect of New Versions.

Once Covered Code has been published under a particular version of the License, You may always continue to use it under the terms of that version. You may also choose to use such Covered Code under the terms of any subsequent version of the License published by Netscape. No one other than Netscape has the right to modify the terms applicable to Covered Code created under this License.

6.3. Derivative Works.

If You create or use a modified version of this License (which you may only do in order to apply it to code which is not already Covered Code governed by this License), You must (a) rename Your license so that the phrases "Mozilla", "MOZILLAPL", "MOZPL", "Netscape", "MPL", "NPL" or any confusingly similar phrase do not appear in your license (except to note that your license differs from this License) and (b) otherwise make it clear that Your version of the license contains terms which differ from the Mozilla Public License and Netscape Public License. (Filling in the name of the Initial Developer, Original Code or Contributor in the notice described in Exhibit A shall not of themselves be deemed to be modifications of this License.)

7. DISCLAIMER OF WARRANTY.

COVERED CODE IS PROVIDED UNDER THIS LICENSE ON AN "AS IS" BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED CODE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED CODE IS WITH YOU. SHOULD ANY COVERED CODE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED CODE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

8. TERMINATION.

8.1. This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. All sublicenses to the Covered Code which are properly granted shall survive any termination of this License. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive.

8.2. If You initiate litigation by asserting a patent infringement claim (excluding declaratory judgment actions) against Initial Developer or a Contributor (the Initial Developer or Contributor against whom You file such action is referred to as "Participant") alleging that:

(a) such Participant’s Contributor Version directly or indirectly infringes any patent, then any and all rights granted by such Participant to You under Sections 2.1 and/or 2.2 of this License shall, upon 60 days notice from Participant terminate prospectively, unless if within 60 days after receipt of notice You either: (i) agree in writing to pay Participant a mutually agreeable reasonable royalty for Your past and future use of
APPENDIX C. THIRD-PARTY SOFTWARE

Modifications made by such Participant, or (ii) withdraw Your litigation claim with respect to the Contributor Version against such Participant. If within 60 days of notice, a reasonable royalty and payment arrangement are not mutually agreed upon in writing by the parties or the litigation claim is not withdrawn, the rights granted by Participant to You under Sections 2.1 and/or 2.2 automatically terminate at the expiration of the 60 day notice period specified above.

(b) any software, hardware, or device, other than such Participant’s Contributor Version, directly or indirectly infringes any patent, then any rights granted to You by such Participant under Sections 2.1(b) and 2.2(b) are revoked effective as of the date You first made, used, sold, distributed, or had made, Modifications made by that Participant.

8.3. If You assert a patent infringement claim against Participant alleging that such Participant’s Contributor Version directly or indirectly infringes any patent where such claim is resolved (such as by license or settlement) prior to the initiation of patent infringement litigation, then the reasonable value of the licenses granted by such Participant under Sections 2.1 or 2.2 shall be taken into account in determining the amount or value of any payment or license.

8.4. In the event of termination under Sections 8.1 or 8.2 above, all end user license agreements (excluding distributors and resellers) which have been validly granted by You or any distributor hereunder prior to termination shall survive termination.

9. LIMITATION OF LIABILITY.

UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL YOU, THE INITIAL DEVELOPER, ANY OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED CODE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES, BE LIABLE TO ANY PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH PARTY’S NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

10. U.S. GOVERNMENT END USERS.


11. MISCELLANEOUS.

This License represents the complete agreement concerning subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This License shall be governed by California law provisions (except to the extent applicable
law, if any, provides otherwise), excluding its conflict-of-law provisions. With respect to disputes in which at least one party is a citizen of, or an entity chartered or registered to do business in the United States of America, any litigation relating to this License shall be subject to the jurisdiction of the Federal Courts of the Northern District of California, with venue lying in Santa Clara County, California, with the losing party responsible for costs, including without limitation, court costs and reasonable attorneys’ fees and expenses. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not apply to this License.

12. RESPONSIBILITY FOR CLAIMS.

As between Initial Developer and the Contributors, each party is responsible for claims and damages arising, directly or indirectly, out of its utilization of rights under this License and You agree to work with Initial Developer and Contributors to distribute such responsibility on an equitable basis. Nothing herein is intended or shall be deemed to constitute any admission of liability.

13. MULTIPLE-LICENSED CODE.

Initial Developer may designate portions of the Covered Code as "Multiple-Licensed". "Multiple-Licensed" means that the Initial Developer permits you to utilize portions of the Covered Code under Your choice of the NPL or the alternative licenses, if any, specified by the Initial Developer in the file described in Exhibit A.

EXHIBIT A -Mozilla Public License.

"The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.mozilla.org/MPL/. Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License. The Original Code is ———————. The Initial Developer of the Original Code is ———————. Portions created by ——————— are Copyright (C) ——— ———————-. All Rights Reserved.
Contributor(s): ————————————–. Alternatively, the contents of this file may be used under the terms of the —— license (the "[—] License"), in which case the provisions of [——] License are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of the [—] License and not to allow others to use your version of this file under the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the [—] License. If you do not delete the provisions above, a recipient may use your version of this file under either the MPL or the [—] License.”

[NOTE: The text of this Exhibit A may differ slightly from the text of the notices in the Source Code files of the Original Code. You should use the text of this Exhibit A rather than the text found in the Original Code Source Code for Your Modifications.]

C.23 UnixUtils

Windows versions of Webinator may include the UnixUtils package of Unix utilities ported to Windows, for use in debugging, analyzing and reporting problems, which includes the following disclaimer:
Disclaimer for UnixUtils

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Karl M. Syring

C.24 PuTTY

Windows versions of Webinator may include the PuTTY SSH connectivity tools, for use in debugging, analyzing and reporting problems, which includes the following license:

PuTTY is copyright 1997-2011 Simon Tatham.

Portions copyright Robert de Bath, Joris van Rantwijk, Delian Delchev, Andreas Schultz, Jeroen Massar, Wez Furlong, Nicolas Barry, Justin Bradford, Ben Harris, Malcolm Smith, Ahmad Khalifa, Markus Kuhn, Colin Watson, and CORE SDI S.A.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
Unix versions of Webinator may include the MIT Kerberos library for use in Negotiate HTTP authentication, which includes the following license:

Copyright (C) 1985-2015 by the Massachusetts Institute of Technology.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Downloading of this software may constitute an export of cryptographic software from the United States of America that is subject to the United States Export Administration Regulations (EAR), 15 CFR 730-774. Additional laws or regulations may apply. It is the responsibility of the person or entity contemplating export to comply with all applicable export laws and regulations, including obtaining any required license from the U.S. government.

The U.S. government prohibits export of encryption source code to certain countries and individuals, including, but not limited to, the countries of Cuba, Iran, North Korea, Sudan, Syria, and residents and nationals of those countries.

Documentation components of this software distribution are licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License. (http://creativecommons.org/licenses/by-sa/3.0/)

Individual source code files are copyright MIT, Cygnus Support, Novell, OpenVision Technologies, Oracle, Red Hat, Sun Microsystems,
FundsXpress, and others.

Project Athena, Athena, Athena MUSE, Discuss, Hesiod, Kerberos, Moira, and Zephyr are trademarks of the Massachusetts Institute of Technology (MIT). No commercial use of these trademarks may be made without prior written permission of MIT.

"Commercial use" means use of a name in a product or other for-profit manner. It does NOT prevent a commercial firm from referring to the MIT trademarks in order to convey information (although in doing so, recognition of their trademark status should be given).

The following copyright and permission notice applies to the OpenVision Kerberos Administration system located in "kadmin/create", "kadmin/dbutil", "kadmin/passwd", "kadmin/server", "lib/kadm5", and portions of "lib/rpc":


WARNING: Retrieving the OpenVision Kerberos Administration system source code, as described below, indicates your acceptance of the following terms. If you do not agree to the following terms, do not retrieve the OpenVision Kerberos administration system.

You may freely use and distribute the Source Code and Object Code compiled from it, with or without modification, but this Source Code is provided to you "AS IS" EXCLUSIVE OF ANY WARRANTY, INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, OR ANY OTHER WARRANTY, WHETHER EXPRESS OR IMPLIED. IN NO EVENT WILL OPENVISION HAVE ANY LIABILITY FOR ANY LOST PROFITS, LOSS OF DATA OR COSTS OF PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, OR FOR ANY SPECIAL, INDIRECT, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THIS AGREEMENT, INCLUDING, WITHOUT LIMITATION, THOSE RESULTING FROM THE USE OF THE SOURCE CODE, OR THE FAILURE OF THE SOURCE CODE TO PERFORM, OR FOR ANY OTHER REASON.

OpenVision retains all copyrights in the donated Source Code. OpenVision also retains copyright to derivative works of the Source Code, whether created by OpenVision or by a third party. The OpenVision copyright notice must be preserved if derivative works are made based on the donated Source Code.

OpenVision Technologies, Inc. has donated this Kerberos Administration system to MIT for inclusion in the standard Kerberos 5 distribution. This donation underscores our commitment to continuing Kerberos technology development and our gratitude for the valuable work which has been performed by MIT and the Kerberos community.
C.25. MIT KERBEROS

Portions contributed by Matt Crawford "crawdad@fnal.gov" were work performed at Fermi National Accelerator Laboratory, which is operated by Universities Research Association, Inc., under contract DE-AC02-76CH03000 with the U.S. Department of Energy.

Portions of "src/lib/crypto" have the following copyright:

Copyright (C) 1998 by the FundsXpress, INC.

All rights reserved.

Export of this software from the United States of America may require a specific license from the United States Government. It is the responsibility of any person or organization contemplating export to obtain such a license before exporting.

WITHIN THAT CONSTRAINT, permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of FundsXpress not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. FundsXpress makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

THIS SOFTWARE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The implementation of the AES encryption algorithm in "src/lib/crypto/builtin/aes" has the following copyright:

Copyright (C) 2001, Dr Brian Gladman "brg@gladman.uk.net", Worcester, UK.

All rights reserved.

LICENSE TERMS

The free distribution and use of this software in both source and binary form is allowed (with or without changes) provided that:

1. distributions of this source code include the above copyright notice, this list of conditions and the following disclaimer;
2. distributions in binary form include the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other associated materials;

3. the copyright holder’s name is not used to endorse products built using this software without specific written permission.

DISCLAIMER

This software is provided ‘as is’ with no explicit or implied warranties in respect of any properties, including, but not limited to, correctness and fitness for purpose.

=====================================================================

Portions contributed by Red Hat, including the pre-authentication plug-in framework and the NSS crypto implementation, contain the following copyright:

Copyright (C) 2006 Red Hat, Inc.
Portions copyright (C) 2006 Massachusetts Institute of Technology All Rights Reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

* Neither the name of Red Hat, Inc., nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
The bundled verto source code is subject to the following license:

Copyright 2011 Red Hat, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

The MS-KKDCP client implementation has the following copyright:

Copyright 2013, 2014 Red Hat, Inc.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The implementations of GSSAPI mechglue in GSSAPI-SPNEGO in "src/lib/gssapi", including the following files:

lib/gssapi/generic/gssapi_err_generic.et
lib/gssapi/mechglue/g_accept_sec_context.c
lib/gssapi/mechglue/g_acquire_cred.c
lib/gssapi/mechglue/g_canon_name.c
lib/gssapi/mechglue/g_compare_name.c
lib/gssapi/mechglue/g_context_time.c
lib/gssapi/mechglue/g_delete_sec_context.c
lib/gssapi/mechglue/g_dsp_name.c
lib/gssapi/mechglue/g_dsp_status.c
lib/gssapi/mechglue/g_dup_name.c
lib/gssapi/mechglue/g_exp_sec_context.c
lib/gssapi/mechglue/g_export_name.c
lib/gssapi/mechglue/g_glue.c
lib/gssapi/mechglue/g_imp_name.c
lib/gssapi/mechglue/g_imp_sec_context.c
lib/gssapi/mechglue/g_init_sec_context.c
lib/gssapi/mechglue/g_initialize.c
lib/gssapi/mechglue/g_inquire_context.c
lib/gssapi/mechglue/g_inquire_cred.c
lib/gssapi/mechglue/g_inquire_names.c
lib/gssapi/mechglue/g_process_context.c
lib/gssapi/mechglue/g_rel_buffer.c
lib/gssapi/mechglue/g_rel_cred.c
lib/gssapi/mechglue/g_rel_name.c
lib/gssapi/mechglue/g_rel_oid_set.c
lib/gssapi/mechglue/g_seal.c
lib/gssapi/mechglue/g_sign.c
lib/gssapi/mechglue/g_store_cred.c
lib/gssapi/mechglue/g_unseal.c
lib/gssapi/mechglue/g_userok.c
lib/gssapi/mechglue/g_utils.c
lib/gssapi/mechglue/g_verify.c
lib/gssapi/mechglue/gssd_pname_to_uid.c
lib/gssapi/mechglue/mglueP.h
lib/gssapi/mechglue/oid_ops.c
lib/gssapi/spnego/gssapiP_spnego.h
lib/gssapi/spnego/spnego_mech.c

and the initial implementation of incremental propagation, including the following new or changed files:
include/iprop_hdr.h
kadmin/server/ipropd_svc.c
lib/kdb/iprop.x
lib/kdb/kdb_convert.c
lib/kdb/kdb_log.c
lib/kdb/kdb_log.h
lib/krb5/error_tables/kdb5_err.et
slave/kpropd_rpc.c
slave/kproplog.c

are subject to the following license:

Copyright (C) 2004 Sun Microsystems, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

==============================================================================

Kerberos V5 includes documentation and software developed at the University of California at Berkeley, which includes this copyright notice:

Copyright (C) 1983 Regents of the University of California.  
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following
disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

================================================================================
Portions contributed by Novell, Inc., including the LDAP database backend, are subject to the following license:

Copyright (C) 2004-2005, Novell, Inc.
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

* The copyright holder’s name is not used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

==================================================================
Portions funded by Sandia National Laboratory and developed by the University of Michigan’s Center for Information Technology Integration, including the PKINIT implementation, are subject to the following license:

COPYRIGHT (C) 2006-2007
THE REGENTS OF THE UNIVERSITY OF MICHIGAN
ALL RIGHTS RESERVED

Permission is granted to use, copy, create derivative works and redistribute this software and such derivative works for any purpose, so long as the name of The University of Michigan is not used in any advertising or publicity pertaining to the use of distribution of this software without specific, written prior authorization. If the above copyright notice or any other identification of the University of Michigan is included in any copy of any portion of this software, then the disclaimer below must also be included.

THIS SOFTWARE IS PROVIDED AS IS, WITHOUT REPRESENTATION FROM THE UNIVERSITY OF MICHIGAN AS TO ITS FITNESS FOR ANY PURPOSE, AND WITHOUT WARRANTY BY THE UNIVERSITY OF MICHIGAN OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE REGENTS OF THE UNIVERSITY OF MICHIGAN SHALL NOT BE LIABLE FOR ANY DAMAGES, INCLUDING SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WITH RESPECT TO ANY CLAIM ARISING OUT OF OR IN CONNECTION WITH THE USE OF THE SOFTWARE, EVEN IF IT HAS BEEN OR IS HEREAFTER ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

==================================================================

The pkcs11.h file included in the PKINIT code has the following license:

Copyright 2006 g10 Code GmbH
Copyright 2006 Andreas Jellinghaus

This file is free software; as a special exception the author gives unlimited permission to copy and/or distribute it, with or without modifications, as long as this notice is preserved.

This file is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY, to the extent permitted by law; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
POURPOSE.

=================================================================================================

Portions contributed by Apple Inc. are subject to the following license:

Copyright 2004-2008 Apple Inc. All Rights Reserved.

Export of this software from the United States of America may require a specific license from the United States Government. It is the responsibility of any person or organization contemplating export to obtain such a license before exporting.

WITHIN THAT CONSTRAINT, permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of Apple Inc. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. Apple Inc. makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

THIS SOFTWARE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

=================================================================================================

The implementations of UTF-8 string handling in src/util/support and src/lib/krb5/unicode are subject to the following copyright and permission notice:

The OpenLDAP Public License
Version 2.8, 17 August 2003

Redistribution and use of this software and associated documentation ("Software"), with or without modification, are permitted provided that the following conditions are met:

1. Redistributions in source form must retain copyright statements and notices,

2. Redistributions in binary form must reproduce applicable copyright statements and notices, this list of conditions, and the following disclaimer in the documentation and/or other materials provided with the distribution, and

3. Redistributions must contain a verbatim copy of this document.
The OpenLDAP Foundation may revise this license from time to time. Each revision is distinguished by a version number. You may use this Software under terms of this license revision or under the terms of any subsequent revision of the license.

THIS SOFTWARE IS PROVIDED BY THE OPENLDAP FOUNDATION AND ITS CONTRIBUTORS "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OPENLDAP FOUNDATION, ITS CONTRIBUTORS, OR THE AUTHOR(S) OR OWNER(S) OF THE SOFTWARE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The names of the authors and copyright holders must not be used in advertising or otherwise to promote the sale, use or other dealing in this Software without specific, written prior permission. Title to copyright in this Software shall at all times remain with copyright holders.

OpenLDAP is a registered trademark of the OpenLDAP Foundation.

Copyright 1999-2003 The OpenLDAP Foundation, Redwood City, California, USA. All Rights Reserved. Permission to copy and distribute verbatim copies of this document is granted.

==================================================================
Marked test programs in src/lib/krb5/krb have the following copyright:

Copyright (C) 2006 Kungliga Tekniska Hgskola
(Royal Institute of Technology, Stockholm, Sweden).
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of KTH nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY KTH AND ITS CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL KTH OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The KCM Mach RPC definition file used on OS X has the following copyright:

Copyright (C) 2009 Kungliga Tekniska Högskola
(Royal Institute of Technology, Stockholm, Sweden).
All rights reserved.

Portions Copyright (C) 2009 Apple Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the Institute nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE INSTITUTE AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE INSTITUTE OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

==================================================================================================================

Portions of the RPC implementation in src/lib/rpc and src/include/gssrpc have the following copyright and permission notice:

Copyright (C) 2010, Oracle America, Inc.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the "Oracle America, Inc." nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

==================================================================================================================

Copyright (C) 2006,2007,2009 NTT (Nippon Telegraph and Telephone Corporation). All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions
APPENDIX C. THIRD-PARTY SOFTWARE

are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer as the first lines of this file unmodified.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY NTT "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL NTT BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

==============================================================================
Copyright 2000 by Carnegie Mellon University
All Rights Reserved

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of Carnegie Mellon University not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

CARNEGIE MELLON UNIVERSITY DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL CARNEGIE MELLON UNIVERSITY BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

==============================================================================
Copyright (C) 2002 Naval Research Laboratory (NRL/CCS)

Permission to use, copy, modify and distribute this software and
its documentation is hereby granted, provided that both the copyright notice and this permission notice appear in all copies of the software, derivative works or modified versions, and any portions thereof.

NRL allows free use of this software in its "As Is" condition and disclaims any liability of any kind for any damages whatsoever resulting from the use of this software.

=================================================================================================

Portions extracted from Internet RFCs have the following copyright notice:

Copyright (C) The Internet Society (2006).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "As Is" basis and the contributor, the organization he/she represents or is sponsored by (if any), the Internet Society and the Internet Engineering Task Force disclaim all warranties, express or implied, including but not limited to any warranty that the use of the information herein will not infringe any rights or any implied warranties of merchantability or fitness for a particular purpose.

=================================================================================================


Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation. Cygnus Support makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

=================================================================================================

Copyright (C) 2006 Secure Endpoints Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

=====================================================================

Portions of the implementation of the Fortuna-like PRNG are subject to the following notice:

Copyright (C) 2005 Marko Kreen
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright (C) 1994 by the University of Southern California

EXPORT OF THIS SOFTWARE from the United States of America may require a specific license from the United States Government. It is the responsibility of any person or organization
contemplating export to obtain such a license before exporting.

WITHIN THAT CONSTRAINT, permission to copy, modify, and distribute this software and its documentation in source and binary forms is hereby granted, provided that any documentation or other materials related to such distribution or use acknowledge that the software was developed by the University of Southern California.

DISCLAIMER OF WARRANTY. THIS SOFTWARE IS PROVIDED "AS IS". The University of Southern California MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. By way of example, but not limitation, the University of Southern California MAKES NO REPRESENTATIONS OR WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. The University of Southern California shall not be held liable for any liability nor for any direct, indirect, or consequential damages with respect to any claim by the user or distributor of the ksu software.

Copyright (C) 1995
The President and Fellows of Harvard University

This code is derived from software contributed to Harvard by Jeremy Rassen.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. All advertising materials mentioning features or use of this software must display the following acknowledgement:

   This product includes software developed by the University of California, Berkeley and its contributors.

4. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS
OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
or TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.

===================================================

Copyright (C) 2008 by the Massachusetts Institute of Technology.
Copyright 1995 by Richard P. Basch. All Rights Reserved.
Copyright 1995 by Lehman Brothers, Inc. All Rights Reserved.

Export of this software from the United States of America may
require a specific license from the United States Government. It
is the responsibility of any person or organization
contemplating export to obtain such a license before exporting.

WITHIN THAT CONSTRAINT, permission to use, copy, modify, and
distribute this software and its documentation for any purpose and
without fee is hereby granted, provided that the above copyright
notice appear in all copies and that both that copyright notice and
this permission notice appear in supporting documentation, and that
the name of Richard P. Basch, Lehman Brothers and M.I.T. not be
used in advertising or publicity pertaining to distribution of the
software without specific, written prior permission. Richard P.
Basch, Lehman Brothers and M.I.T. make no representations about the
suitability of this software for any purpose. It is provided "as
is" without express or implied warranty.

===================================================

The following notice applies to "src/lib/krb5/krb/strptime.c" and
"src/include/k5-queue.h".

Copyright (C) 1997, 1998 The NetBSD Foundation, Inc.
All rights reserved.

This code was contributed to The NetBSD Foundation by Klaus Klein.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above
   copyright notice, this list of conditions and the following
   disclaimer in the documentation and/or other materials provided
with the distribution.

3. All advertising materials mentioning features or use of this software must display the following acknowledgement:

   This product includes software developed by the NetBSD Foundation, Inc. and its contributors.

4. Neither the name of The NetBSD Foundation nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

   THIS SOFTWARE IS PROVIDED BY THE NETBSD FOUNDATION, INC. AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE FOUNDATION OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

=================================================================

The following notice applies to Unicode library files in "src/lib/krb5/unicode":

   Copyright 1997, 1998, 1999 Computing Research Labs,
   New Mexico State University

   Permission is hereby granted, free of charge, to any person
   obtaining a copy of this software and associated documentation
   files (the "Software"), to deal in the Software without
   restriction, including without limitation the rights to use, copy,
   modify, merge, publish, distribute, sublicense, and/or sell copies
   of the Software, and to permit persons to whom the Software is
   furnished to do so, subject to the following conditions:

   The above copyright notice and this permission notice shall be
   included in all copies or substantial portions of the Software.

   THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
   EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
   MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
   NONINFRINGEMENT.  IN NO EVENT SHALL THE COMPUTING RESEARCH LAB OR
   NEW MEXICO STATE UNIVERSITY BE LIABLE FOR ANY CLAIM, DAMAGES OR
   OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
   OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE
The following notice applies to "src/util/support/strncpy.c":

Copyright (C) 1998 Todd C. Miller "Todd.Miller@courtesan.com"

Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

The following notice applies to "src/util/profile/argv_parse.c" and "src/util/profile/argv_parse.h":

Copyright 1999 by Theodore Ts’o.

Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies. THE SOFTWARE IS PROVIDED "AS IS" AND THEODORE TS’O (THE AUTHOR) DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE. (Isn’t it sick that the U.S. culture of lawsuit-happy lawyers requires this kind of disclaimer?)

The following notice applies to SWIG-generated code in "src/util/profile/profile_tcl.c":

Copyright (C) 1999-2000, The University of Chicago

This file may be freely redistributed without license or fee provided this copyright message remains intact.
The following notice applies to portions of "src/lib/rpc" and "src/include/gssrpc":

Copyright (C) 2000 The Regents of the University of Michigan. All rights reserved.

Copyright (C) 2000 Dug Song "dugsong@UMICH.EDU". All rights reserved, all wrongs reversed.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Implementations of the MD4 algorithm are subject to the following notice:

Copyright (C) 1990, RSA Data Security, Inc. All rights reserved.

License to copy and use this software is granted provided that it is identified as the "RSA Data Security, Inc. MD4 Message Digest Algorithm" in all material mentioning or referencing this software or this function.
License is also granted to make and use derivative works provided that such works are identified as "derived from the RSA Data Security, Inc. MD4 Message Digest Algorithm" in all material mentioning or referencing the derived work.

RSA Data Security, Inc. makes no representations concerning either the merchantability of this software or the suitability of this software for any particular purpose. It is provided "as is" without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this documentation and/or software.

================================================================================================

Implementations of the MD5 algorithm are subject to the following notice:

Copyright (C) 1990, RSA Data Security, Inc. All rights reserved.

License to copy and use this software is granted provided that it is identified as the "RSA Data Security, Inc. MD5 Message-Digest Algorithm" in all material mentioning or referencing this software or this function.

License is also granted to make and use derivative works provided that such works are identified as "derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm" in all material mentioning or referencing the derived work.

RSA Data Security, Inc. makes no representations concerning either the merchantability of this software or the suitability of this software for any particular purpose. It is provided "as is" without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this documentation and/or software.

================================================================================================

The following notice applies to
"src/lib/crypto/crypto_tests/t_mddriver.c":

Copyright (C) 1990-2, RSA Data Security, Inc. Created 1990. All rights reserved.

RSA Data Security, Inc. makes no representations concerning either the merchantability of this software or the suitability of this software for any particular purpose. It is provided "as is" without express or implied warranty of any kind.
These notices must be retained in any copies of any part of this documentation and/or software.

===================================================
Portions of "src/lib/krb5" are subject to the following notice:

Copyright (C) 1994 CyberSAFE Corporation.
All Rights Reserved.

Export of this software from the United States of America may require a specific license from the United States Government. It is the responsibility of any person or organization contemplating export to obtain such a license before exporting.

WITHIN THAT CONSTRAINT, permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of M.I.T. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. Furthermore if you modify this software you must label your software as modified software and not distribute it in such a fashion that it might be confused with the original M.I.T. software. Neither M.I.T., the Open Computing Security Group, nor CyberSAFE Corporation make any representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

===================================================
Portions contributed by PADL Software are subject to the following license:

Copyright (c) 2011, PADL Software Pty Ltd. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of PADL Software nor the names of its contributors may be used to endorse or promote products derived
from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY PADL SOFTWARE AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL PADL SOFTWARE
OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
special, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.

================================================================================

The bundled libev source code is subject to the following license:

   All files in libev are Copyright (C)2007,2008,2009 Marc Alexander
   Lehmann.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

* Redistributions of source code must retain the above copyright
  notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright
  notice, this list of conditions and the following disclaimer in
  the documentation and/or other materials provided with the
distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRIC LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
OF THE POSSIBILITY OF SUCH DAMAGE.

Alternatively, the contents of this package may be used under the
terms of the GNU General Public License ("GPL") version 2 or any
later version, in which case the provisions of the GPL are
applicable instead of the above. If you wish to allow the use of
your version of this package only under the terms of the GPL and
not to allow others to use your version of this file under the BSD license, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL in this and the other files of this package. If you do not delete the provisions above, a recipient may use your version of this file under either the BSD or the GPL.

Files copied from the Intel AESNI Sample Library are subject to the following license:

Copyright (C) 2010, Intel Corporation
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

* Neither the name of Intel Corporation nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

C.26  Cyrus SASL

Unix versions of Webinator may include the Cyrus SASL library for use in Negotiate HTTP authentication, which includes the following license:
Copyright (c) 1998-2003 Carnegie Mellon University. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The name "Carnegie Mellon University" must not be used to endorse or promote products derived from this software without prior written permission. For permission or any other legal details, please contact Office of Technology Transfer Carnegie Mellon University 5000 Forbes Avenue Pittsburgh, PA 15213-3890 (412) 268-4387, fax: (412) 268-7395 tech-transfer@andrew.cmu.edu

4. Redistributions of any form whatsoever must retain the following acknowledgment:
"This product includes software developed by Computing Services at Carnegie Mellon University (http://www.cmu.edu/computing/)."

CARNEGIE MELLON UNIVERSITY DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL CARNEGIE MELLON UNIVERSITY BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
Appendix D

Texis Version 6 Features and Changes

Texis version 6 introduced many new features and enhancements. Some existing features were also modified to have different behavior. The following is a discussion of changes from Texis version 5 to 6, starting with a general overview of important changes, loosely grouped by functionality. All changes are then discussed in more detail in the sections that follow.

Changes to be especially aware of during upgrading are noted with Caveat and are in bold text. Also note that all changes here refer to the Texis version, which is not necessarily the version of the main Thunderstone product (which may be built on top of Texis). The Texis version can be obtained by running texis -version on the command line, or through the Maintenance – Thunderstone Information web menu (if applicable).

D.1 Overview

D.1.1 Query Handling

Metamorph queries in version 6 are handled more seamlessly from a user’s perspective. Words are case-insensitive across the entire Unicode 5.1 locale-independent character set, not just the limited or non-existent 8-bit locale supported by the local operating system. UTF-8 is supported directly, and does not need to be translated to an 8-bit character set for searching. Accents, diacritical marks (umlauts etc.) and full-width vs. half-width differences are ignored by default, and ligatures are expanded. All of these features are tunable; see the <apicp textsearchmode> setting (p. 121) for example.

Wildcards with a suffix bind to a single word as expected in version 6. For example, the query “st*ion” matches “stallion” but not “stuff an onion” nor “stationary”; see the SQL settings wildoneword and wildsufmatch for details. A within-operator query (w/N) counts words instead of characters, and the count is the overall length of the match, not a left-and-right radius from an arbitrary term. Thus searching for “clinton bush obama w/10” will find those terms within a single 10-word span. See the <apicp withinmode> setting (p. 124) for details.
D.1.2 Search Results Improvements

Abstracts (the document summaries provided in search results) are improved in version 6 with support for multiple “snippets”. This means that the abstract may contain more than one chunk from the document (separated by ellipses), if the search terms are widely separated. This increases the likelihood that all search terms will be highlighted in the abstract. See the Vortex <abstract> function (p. 270) for details.

Hit highlighting was upgraded with per-term and whole-query CSS support. Each term is highlighted in a different color, and all the formats can be customized with HTML style sheets; see <fmt> hit markup sub-codes I/C (p. 104) and <fmtcp querystyle> etc. settings (p. 108) for details.

In Webinator, administrators gained more flexibility with the addition of meta searches, which allow multiple profiles to be searched as one. This can be useful for large/complex sites, where different crawl rules are needed for different parts, or when search users demand more fine-tuned sectioning that is available with standard URL-based Categories alone.

A Maintenance section adds new admin support tools such as thesaurus (synonym) customization, GUI application of new licenses, network testing, and off-site backup of profile settings. Search results are available in XML format as well as via a SOAP API, for easier integration with other applications.

Support for searching protected or multi-level-access sites was added, with results authorization: users can be limited to only those search results they have access to. Search results can also be grouped by site, to present a greater variety of top-level sites in the result set when the majority of results are from only one or two sites. Searching of CJK (Chinese-Japanese-Korean) languages was improved with the addition of the Language Analysis Module (available separately).

D.1.3 URL Fetching and Web Server Changes

Version 6 introduced some new URL fetch settings and behaviors. HTTP/1.1 is now supported by default, including gzip and other compression formats (see <urlcp encodings> p. 202). A new form of secure SSL transfers is supported in <fetch>, the OPTIONS/Upgrade method, which (for servers that support it) allows HTTP transactions to be SSL-secure without the need for a separate HTTPS server and namespace (see <urlcp secure>, p. 212). FTP logins are supported more fully with login-dir relative paths, so that a user directories may be accessed easier via FTP: there is no need to specify the home directory in the URL (see <urlcp ftprelativepaths> p. 204). Uncommon or misspelled character sets can be mapped to their correct equivalents with a charset config file, avoiding some “Unknown character set” errors (see <urlcp charsetconfigfromfile> p. 217).

The vhttpd web server configuration file supports some %-variables for greater clarity. For example, %SERVERROOT% may be used in a setting value to indicate the web server root path. The AllowExt and ExcludeExt (p. 590) defaults changed to allow all files except for Vortex source and object files, so that new file types do not require constantly editing the config file to add then to AllowExt. The Texis Monitor web server (but not vhttpd) supports SSL; see SSL Engine et al. settings in the [Httpd] section of texis.ini.
D.1.4 Filename Changes

Some filenames and paths were changed in Texis version 6. The Texis config file \texttt{texis.cnf} (in the install dir) was renamed to \texttt{texis.ini} (now in the \texttt{conf} subdir). This places it in the same dir as other config files, and gives it an extension that is more appropriate for its Windows-style config syntax. The old location is still supported as a fallback.

Vortex source files have the extension \texttt{.vs} in version 6, instead of no extension. No-extension source files are still supported for back compatibility, and the extension need not be specified on the command line or URL. The file extension allows easier searching and mapping of Vortex files. The source extension is changeable (not recommended) in \texttt{texis.ini} via \texttt{[Texis] Vortex Source Extensions} (p. 577).

The Vortex \texttt{ErrorScript} and \texttt{ErrorFile} defaults (p. 576) are set differently, though the end result is the same. In version 6, the default uses the new \texttt{%SCRIPTROOT\%} variable, which is not available in earlier versions. However the net location – \texttt{texis/scripts} subdir of the install dir – is the same as previous versions.

D.1.5 Programming Features

Other changes in Texis version 6 primarily affect Vortex and SQL programmers. The default APICP settings for \texttt{tsql} are the same as for \texttt{texis} (Vortex) in version 6, to avoid confusion when testing Vortex SQL via command-line \texttt{tsql}. This means query protection is \texttt{on} by default in \texttt{tsql}. System-wide defaults for APICP settings can now be set in \texttt{texis.ini} (see the \texttt{[Apicp]} section). The APICP setting/alias \texttt{querysettings} was added (p. 116); it replaces the deprecated (but still temporarily supported) \texttt{texisdefaults} setting, which is a bit of a misnomer. An \texttt{<apiinfo>} function was added to Vortex (p. 125), to obtain the current value of \texttt{<apicp>} settings.

A new XML and XSLT API was added, which offers more flexible parsing of complex or multi-level documents than possible with a flat \texttt{timport}. XSL stylesheets can be applied to XML documents with the XSLT API. See the XML API section (p. 336) for details.

The handling of \texttt{strlst} and other multi-value SQL types to and from SQL statements in Vortex was changed with the addition of \texttt{<sqlcp arrayconvert>} (p. 127). In version 6, multi-value Vortex parameters passed to a SQL statement become \texttt{strlst} instead of comma-separated string parameters, and returned SQL columns of type \texttt{strlst} are automatically broken up into multi-value Vortex variable arrays.

Columns returned from SQL \texttt{INSERT}, \texttt{DELETE} and \texttt{UPDATE} statements can be limited with the same flexibility as those from \texttt{SELECT} statements, with the addition of the \texttt{okvars} option to the \texttt{<sql>} statement (p. 27). This limits the columns from SQL that are actually returned as Vortex variables, to only those listed in \texttt{okvars}. This can be useful in situations where the new return value from just one column in an \texttt{UPDATE} statement is desired, without “stomping” other Vortex variables for the other columns.

New SQL types \texttt{int64} and \texttt{uint64} were added, for 64-bit signed and unsigned integers. These are valid across all platforms, unlike \texttt{long} whose size varies by platform. Some Vortex functions that return large integer values may use \texttt{int64} in version 6, e.g. \texttt{$\text{ret.size}} from \texttt{<stat>}. The \texttt{varint} and \texttt{int(N)} types were also added, for declaring multi-value integers.

Two new columns were added to the “tables” produced by direct \texttt{indexaccess SELECTing} from
Metamorph indexes: RowCount and OccurrenceCount. The latter – for Metamorph inverted indexes only – gives the count of all hits in all rows, not just the number of rows, for the word.

More result set size information is available from <sql> statements with the $sqlresult... variables added in version 6 (p. 34). These provide more accurate and detailed information than $rows.min, $rows.max and $indexcount. For example, both the number of returned as well as total matched rows are available from GROUP BY statements.

Debugging Vortex scripts is easier with the $?var and ??var variable printing syntaxes (p. 7). These syntaxes print more details about the Vortex variable var, such as its name, type, and all values, which can be useful when diagnosing unexpected script behavior.

The <exec> statement automatically double-quotes arguments as needed under Windows, usually obviating the need for separate statements for Windows and other OSes. See the quoteargs <exec> flag for details (p. 44). More Vortex functions that output HTML are now XHTML-compliant, e.g. <options>, <checkbox>, <radiobutton> etc.

The REX set-subtract operator “--” was added, for subtracting from a set. E.g. “[\alpha--z]” matches any alphabetic character except “z”. The anytotx plugin produces some extra headers, and supports Open Document Format (ODF) files.

\section*{D.2 Filename Changes}

Some config files and paths were changed in Texis version 6:

\subsection*{D.2.1 texis.cnf Moved - Caveat}

The texis.cnf Texis config file has been moved and renamed to texis.ini in Texis version 6, for co-location with other config files and consistency with other Windows-ini-format files. The version 5 and earlier location will be checked as a secondary fallback.

\textbf{Caveat: The version 5 location is checked only if the version 6 texis.ini file is not found first.}

\subsection*{D.2.2 Default Vortex Source File Extension .vs}

In Texis version 6, the default Vortex source file extension is “.vs” for clarity. Empty (no extension) is still accepted for back-compatibility. The source extension need not be specified on the command line nor URLs. Under Windows, a file mapping is added at installation to map .vs to Wordpad or another editor. The source extension is changeable (not recommended) in texis.ini via [Texis] Vortex Source Extensions (p. 577).

\subsection*{D.2.3 ErrorScript, ErrorFile Initialized Differently - Caveat}

The default ErrorScript and ErrorFile locations in the Texis version 6 texis.ini file (p. 576) use %SCRIPTROOT% in their values, a variable not available in version 5 or earlier. The effective location is
still the same (errorscript and errorfile in the texis/scripts directory).

**Caveat:** if a version 5 or earlier Texis accesses a version 6 or later texis.cnf – manually moved to that old (version 5) location instead of texis.ini – the error “ErrorScript or SERVER_ROOT must be an absolute path” may result.

To revert to version 5 behavior, [Texis] ErrorScript may be set to the version-5-compatible value \%INSTALLDIR\%/texis/scripts/errorscript (and ErrorFile set similarly).

### D.3 SQL and apicp Setting Changes

Several apicp and SQL settings were added or changed in Texis version 6:

#### D.3.1 Unicode `textsearchmode`, `stringcomparemode` - Caveat

Texis Version 6 has improved Unicode (international/foreign/hi-bit/UTF-8) character support. Two new settings were introduced: `textsearchmode` (p. 121) and `stringcomparemode` (p. 120). Both have the same set of possible values, and offer more flexibility in how text searches and string comparisons (respectively) are handled. Some features:

- **Full Unicode case-insensitivity**
  By default, text searches (e.g. the LIKE operator) are case-insensitive in version 6 for the entire Unicode 5.1 locale-independent character set, not just the given operating system’s locale (which may be inconsistent and does not support characters beyond U+00FF).

- **UTF-8 support**
  UTF-8 is the expected character set, though ISO-8859-1 is still accepted. (Other character sets are converted automatically.)

- **Full-width ASCII**
  Full-width ASCII characters (used in CJK contexts) match their normal/half-width ASCII counterparts.

- **Diacritics ignored**
  Diacritic marks – umlauts, accents, etc. – are ignored, so that e.g. “f¨ur” matches “fur”.

- **Ligatures expanded**
  Ligatures are expanded to match their expanded counterparts, e.g. “œ” (U+0153) will match “oe”.

All of these behaviors can be controlled with the (new in version 6) `textsearchmode` and `stringcomparemode` apicp settings (see the Vortex manual for details).

**Caveat:** A version 5 or earlier Texis should not access or modify a regular (B-tree) or Metamorph index originally created by a version 6 or later Texis, unless `stringcomparemode` was set to `ctype`, `respectcase`, `iso-8859-1` (regular indices) or `textsearchmode` was set to `ctype`, `ignorecase`, `iso-8859-1` (Metamorph indices) at creation. If hi-bit/UTF-8/Unicode characters exist in the data, index corruption may result from Texis 5 modifications.
The `stringcomparemode` setting also affects the functions `<xtree>`, `<strstr>`, `<strtstri>`, `<substr>`, `<strcmp>`, `<strcmpi>`, `<strncmp>`, `<strlen>`, `<strrev>`, `<upper>`, `<lower>`, `<sort>`, `<uniq>`, `upper()`, `lower()`, `initcap()`, `text2mm()` and `length()`. The `length()`/`strlen()` functions count charset characters (e.g. UTF-8 characters) not bytes.

Version 5 and earlier behavior can be restored by default by setting the `texis.ini` setting `[Apicp] Text Search Mode` to `ctype`, `ignorecase`, `iso-8859-1`, and `[Apicp] String Compare Mode` to `ctype`, `respectcase`, `iso-8859-1`.

D.3.2 `tsql` `apicp` Defaults Same as Vortex

In Texis version 6, `apicp` (and some related SQL settings’) defaults in `tsql` are the same as Vortex, for consistency. This means query protection is enabled, e.g. linear Metamorph searches are not allowed, `allineardict` is off, etc.

To restore version 5 behavior by default (query protection off), use the SQL `set querysettings='texis5defaults'` (and set non-`apicp` settings as needed, e.g. `allineardict` on).

D.3.3 `apicp` Defaults Settable in `texis.ini`

Defaults for `apicp` settings can be set in Texis version 6, in the `[Apicp]` section of `texis.ini`. These defaults are used by `tsql` and `texis` (Vortex).

D.3.4 `texisdefaults` Deprecated - Caveat

In Texis version 6, `<apicp texisdefaults>` is still supported but deprecated. Use instead the setting `<apicp querysettings texis5defaults>` (p. 116). `texisdefaults` is deprecated because it is ambiguous, since the Texis (i.e. `tsql`) defaults have changed in version 6.

The warning message when using `texisdefaults` can be suppressed with the `texis.ini` setting `[Texis] Texit Default Warning set to off` (p. 579).

*Caveat: The `texisdefaults` setting may be removed in a future release, and a warning message is issued when it is used.*

D.3.5 `apicp` Setting `querysettings` Added

The setting “alias” `querysettings` (p. 116) was added to `apicp` in Texis version 6. It has several options for controlling groups of `apicp` settings; see the Vortex manual for details.
D.3.6 withinmode Defaults to word span

The withinmode apicp setting (p. 124) defaults to the new mode word span in Texis version 6. This means that a Metamorph search using the w/N operator will by default search for within N words, not characters.

The version 5 behavior can be restored by default by setting the texis.ini setting [Apicp] withinmode to char radius.

D.3.7 wildsingle On by Default

The SQL setting wildsingle defaults to on in version 6. This means that a Metamorph query like “st*ion” matches “stallion” but not “stuff an onion” nor “stationary”.

The version 5 behavior can be restored with the SQL set wildsingle='off'.

D.4 Programming Features

Details of Texis version 6 features that primarily affect Vortex and SQL programmers are discussed below.

D.4.1 Multi-Part Abstracts

The <abstract> function in Vortex (p. 270) is improved in Texis version 6. Multiple “snippets” of a query may now be returned. This means that if a query contains multiple terms, and they only occur widely separated in the source text, multiple sections of the document may be returned (separated by ellipses), to show occurrences of more of the query terms. Previous versions would return a monolithic abstract containing only one of the terms, or perhaps between terms. Leading/trailing ellipses are also added in version 6, if parts of the document before/after the abstract were removed. Handling of file paths and URLs is also improved. New modes querymultiple and querybest were added; see the Vortex manual on <abstract> for details.

The version 5 behavior (monolithic return value) can be obtained with the querysingle option.

D.4.2 Hit Highlighting Enhanced

Query/hit highlighting is improved in Texis version 6. CSS classes and styles are used, and individual terms in the query may be highlighted differently from one another to distinguish them. The Vortex <fmt> markup flags %mI and %mC were added (p. 104), for inline styles and CSS classes respectively. These markup each term in a different background color. The version 5 flag %mb is still fully supported, for bold-only highlighting.

Additionally, the <fmtcp queryfixupmode> setting was added (p. 109), which controls how Metamorph queries are modified when used for hit highlighting. It defaults to findsets in version 6,
which preserves the query (including delimiters) and finds all non-NOT sets. The version 5 default behavior
– append @0 w/. to the query – can be restored with <fmtcp queryfixupmode withindot>.

D.4.3 Multi-value SQL Parameters and strlst Changes, arrayconvert - Caveat

In Texis version 6, <sqlcp arrayconvert> (p. 127) was added, and is on by default. This expands
strlst values from SQL into individual varchar values when returned in Vortex variables, and converts
Vortex multi-value varchar parameters to strlst when passed to SQL (and Metamorph treats strlst
queries as an equiv list). In version 5, multi-value varchar parameters were passed as a varchar
parenthetical list instead (i.e. a Metamorph search was always assumed).

Caveat: This can affect the return values of <sql> statements, and cause returned column variables
to become out-of-sync row wise.

To restore version 5 behavior by default, set the texis.ini setting [Texis] Array Convert to
off.

D.4.4 <sql okvars> Option

The okvars option to <sql> (p. 27) was added in Texis version 6. This allows the returned columns from
<sql> to be filtered, by only allowing the listed variables to be returned to Vortex. This can be useful when
only certain returned columns from INSERT, DELETE or UPDATE statements are desired, without
disturbing existing values for Vortex variables of the other columns.

D.4.5 sqlresult... Variables Returned from <sql>

Several new special Vortex variables were introduced in Texis version 6. The sqlresult... variables
(p. 34) return more information from <sql> calls, and are more accurate than $rows.min and
$rows.max. See the Vortex manual for details.

D.4.6 Debug Syntax for Vortex Variables

In Texis version 6, Vortex variables may be printed with the $?var or $??var syntaxes (p. 7). Both print
the variable name and all of its values; the latter syntax also uses CSS to highlight values more clearly, and
prints the script and line number. These syntaxes can be useful for debugging Vortex scripts.

D.4.7 <exec> Double-quotes Args with Space

In Texis version 6, under Windows, the <exec> function in Vortex automatically double-quotes any
arguments containing whitespace (and not already containing double-quotes), for more correct parsing by
the program. This generally obviates the need to explicitly double-quote such arguments in Vortex. See the
quoteargs flag to <exec> (p. 45).
The version 5 behavior (no automatic quoting) can be restored by default with the `texis.ini` setting `[Texis] Exec Quote Args` (p. 579) set to 0. It can also be set in the flags to individual `<exec>` statements (see the Vortex manual).

**D.4.8 XML, XSLT API Added**

An XML and XSLT parsing API was added in Texpr version 6, the `libxml2` library. See the XML API section (p. 336) of Vortex manual for details.

**D.4.9 int64, uint64 Types Added**

Texpr version 6 added the new SQL types `int64` and `uint64`, for 64-bit signed and unsigned integers, respectively. Several functions and Vortex variables use `int64` to return values in version 6: `<stat>$ret.size`, `<sysinfo physmem>`, `<sysinfo freespace>`, `<sum "provider=odbc"> interface in Vortex supports BIGINT via `int64`. In version 5 and earlier these functions returned `long` or `double` values.

**D.4.10 varint, int(N) Support**

Limited multi-value integer support was added in Texpr version 6, e.g. `varint` and `int(N)` field types can be declared in SQL.

**D.4.11 Post-processing `putmsg` Changes - Caveat**

In Texpr version 6, several Metamorph index search warning messages changed:

- Index expressions only substring-match ‘...’ changed to
  Index expressions match non-prefix substring of term ‘...’
- Term ‘...’ only partially matches index expression(s) changed to
  Index expression(s) only partially prefix-match term ‘...’
- Phrase ‘...’ has no terms matching index expression(s) changed to
  Index expression(s) do not match any terms of phrase ‘...’

*Caveat: Scripts that scan for these messages may need to be updated.*

**D.4.12 More XHTML Compliance**

In Texpr version 6, more HTML functions are XHTML-compliant. The Vortex functions `<options>`, `<radiobutton>`, `<checkbox>`, `<adminsql>`, `<cal>`, `<calendar>` produce XHTML-compliant output, as do the default Vortex error script and `vhttpd` error messages and directory listing.
D.4.13  indexaccess Columns Added

When accessing a Metamorph index as a table, via indexaccess, two new columns were added in version 6. The first is RowCount, the count of table rows (documents) the word occurs in. (This is the same as the Count value, but more specifically-named. The Count column is deprecated and may be removed in a future release.) The other new column is OccurrenceCount, which is the count of all hits of the word, i.e. sum of all occurrences in all documents. OccurrenceCount is available for Metamorph inverted indexes only; for non-inverted metamorph indexes it is not present.

D.4.14  REX Set-Subtract Operator

A new REX operator for set subtraction was added in Texis version 6, the double-dash (“--”). This operator subtracts its right side from its left side. For example, “[\alpha--x]” would match any alphabetic character except “x”. See the Vortex manual on <rex> (p. 148) for details.

D.4.15  Vortex Scheduler Port Changed

The Vortex scheduler server defaults to port 10006 in Texis version 6 – instead of port 10005 – if the texis.ini setting [Scheduler] SSL Engine is set to on. (SSL Engine defaults to optional, so this change normally does not happen. In any event, the Vortex schedule server is normally bound to localhost as well, since it is local-machine usage only.)

D.4.16  More anytotx Headers and Formats

The anytotx program may print additional headers – X-Anytotx-Content-Type-Initial and X-Anytotx-Identified-By-Initial – in Texis version 6. These have the same meaning as their non-Initial counterparts, but the -Initial versions have a lower precedence. E.g. an X-Anytotx-Content-Type header overrides X-Anytotx-Content-Type-Initial if both are present.

anytotx also supports Open Document Format files in version 6.

D.5  New URL Fetching Settings

Several <urlcp> settings were added in Texis version 6:

D.5.1  urlcp secure

The <urlcp secure> setting (p. 212), added in version 6, controls whether transactions should be or must be secure, including support for OPTIONS-Upgrade secure HTTP fetches.
D.5.2 urlcp putmsg

The `<urlcp putmsg>` setting (p. 229), added in version 6, controls caching and disambiguation of `<fetch parallel>` error messages.

D.5.3 urlcp fetchmeter

The `<urlcp fetchmeter>` setting (p. 229), added in version 6, prints a progress meter for fetches.

D.5.4 urlcp httpversion

The `<urlcp httpversion>` setting (p. 205), added in version 6, allows the HTTP version to be changed. HTTP/1.1 is supported by default, and compressed HTML is supported with HTTP/1.1.

D.5.5 urlcp ftprelativepaths

FTP file paths in URLs are login-dir-relative in Texis version 6. For example, the URL ftp://ftp.acme.org/dir/subdir/file will be obtained with the FTP command GET dir/subdir/file, not GET /dir/subdir/file as in previous versions. For most FTP URLs that are accessed with an anonymous chrooted user, this makes no difference. But for non-chrooted accounts (such as real users), this allows fetching of files from user home directories without having to specify them in the URL. An absolute path can still be given by escaping the slash in the URL, e.g. ftp://ftp.acme.org/%2Fdir/subdir/file.

The version 5 behavior (absolute paths) may be restored with `<urlcp ftprelativepaths off>` (p. 204) in scripts.

D.5.6 urlcp sslprotocols

`<urlcp sslprotocols>` (p. 216) was added in Texis version 6; it changes which SSL protocols to use.

D.5.7 urlcp ignoreanchorframes

Boolean setting added in Texis version 6: whether to ignore src="#" frames and IFRAMEs, which are usually redundant from a content perspective. On by default in version 6; turn off for version 5 behavior. See p. 205 for details.

D.5.8 urlcp charsetconfigfromfile

Charset aliases may be configured in Texis version 6 with an optional charset config file, settable with the `urlcp settings charsetconfigfromfile` or `charsetconfigfromtext` (p. 217). This allows
alternate names for known charsets to be supported, or for incorrectly-labelled charsets to be properly interpreted. Also see the texis.ini setting [Texis] Charset Config (p. 578).

D.6 Web Server Changes

Some web server changes were implemented in Texis version 6:

D.6.1 vhttpd AllowExt, ExcludeExt Defaults Changed - Caveat

The vhttpd web server setting AllowExt (p. 590) default in Texis version 6 was changed to “*”, not a fixed list of known extensions. ExcludeExt was changed to “” .vs .vtx”. This avoids the hassle of having to edit AllowExt every time a new document format is added. Restricting Vortex scripts/objects is retained in case ScriptRoot was manually reconfigured to use document root. Normally Vortex scripts are in a separate texis/scripts directory by default anyway.

Caveat: The version 6 AllowExt and DefaultExt defaults allow all file extensions except Vortex scripts/objects (and files with no extension) to be downloaded via vhttpd.

D.6.2 Variables Supported in vhttpd Settings

In Texis version 6, settings in the vhttpd config file conf/vhttpd.conf support a number of builtin variables that may be referenced in setting values. For example, %SERVER_ROOT% may be included in a value for the server root directory. These variables can be used to help clarify setting values.

D.6.3 Monitor Web Server SSL Support

In Texis version 6, the Texis Monitor web server supports SSL/HTTPS. New SSL settings were added to the texis.ini [Httpd] and [Scheduler] sections. The vhttpd web server does not yet support SSL.

D.7 Webinator Changes

Webinator 6, which uses Texis 6, has the following new features and changes. See the Webinator manual for details:

D.7.1 Meta Search

Webinator 6 supports meta-searching of multiple profiles. A profile can be created that does not contain crawl data itself, but instead searches one or more other profiles – even on other machines – and combines the results for homogenous presentation as a single set.
D.7.2 Maintenance Section

Webinator 6 added a Maintenance section to the administration interface, with various information and support tools (e.g. license, network test, etc.).

D.7.3 License Updates Applicable via Web Interface

Texis license updates (license.upd files) can be securely applied via the Webinator web interface in Webinator/Texis version 6, via the Maintenance menu option. There is a new command-line option for applying licenses as well, --apply-license.

This GUI feature can be disabled (for version 5 behavior) by setting the texis.ini setting [License Update] User to nothing (empty).

D.7.4 XML Results

Webinator 6 supports XML output of results, for complete flexibility in results presentation to the user. The visible results can be completely customized with admin-editable XSL stylesheets.

D.7.5 SOAP API

Webinator 6 has a SOAP search API, for automated searches by other software.

D.7.6 Compressed HTML Support

Webinator 6 supports HTTP/1.1 compression, which can improve crawl speed and decrease network bandwidth usage.

D.7.7 Faster Category Searches

Webinator 6 has faster category searches in many environments, via the Categories Type setting.

D.7.8 Customizable Thesaurus

Webinator 6 allows customizable thesauruses to be created, for custom synonyms or phrases. Search terms common to a particular industry or site can thus be added to aid users when searching for a synonymous but different term.
D.7.9 Results Authorization

Webinator 6 supports results authorization, for searching access-restricted sites. Users can be required to enter their existing credentials, and are only permitted to see results visible to those credentials.

D.7.10 Group by Site

Webinator 6 supports group-by-site, which allows results to be grouped by web site. This can increase the variety of sites that are presented in the URLs of a given page of search results, by suppressing later results from sites that are already present on the page.

D.7.11 Backup and Restore Settings

Webinator 6 supports the backup and restore of admin settings. Settings can be downloaded to an XML file for off-site backup, and later restored from XML if needed.

D.7.12 Apply/Revert Appearance Changed to Buttons

The Apply Appearance and Revert Appearance checkboxes under Search Settings have been replaced with easier-to-use Update Test, Update Live and Test, and Copy Live to Test buttons at the bottom of the form.

D.7.13 CSS Admin GUI

The Webinator 6 admin interface look and feel uses CSS.

D.7.14 Text Search Mode

Webinator 6 added the Text Search Mode setting, which controls how accents/diacritics/etc. are ignored for Unicode/hi-bit characters. It supports the same features as the Texis textsearchmode setting.

D.7.15 Language Analysis Module

Webinator 6 added a Language Analysis Module (available separately) for additional processing of CJK (Chinese/Japanese/Korean) language data and searches. Contact Thunderstone for details.

D.7.16 Extensions Renamed and Default Changed

The Webinator/Appliance setting Extensions was renamed to Allow Extensions in Texis version 6 for clarity, and the default is empty (i.e. allow all extensions). This will affect new profiles created with
version 6, not existing profiles.

D.7.17 Exclusions Default Changed

The Webinator/Appliance setting Exclusions default changed in Texit version 6, to just logout. This will affect new profiles created with version 6, not existing profiles.
Appendix E

Texis Version 7 Features and Changes

Texis version 7 introduced some new features and enhancements, as well as altering a few existing features; both are discussed below. Changes to be especially aware of during upgrading are noted with Caveat and/or are in bold text. Also note that “version” here refers to the Texis version, which is not necessarily the version of the main Thunderstone product (which may be built on top of Texis). The Texis version can be obtained by running texis -version on the command line, or through the Maintenance – Thunderstone Information web menu (if applicable).

E.1 Webinator Features - Caveat

No significant Webinator features were introduced with Texis version 7. However, the Webinator 6.2 Vortex scripts have been altered to work with Texis version 7 binaries.

Caveat: Webinator 6.1 and earlier scripts are incompatible with Texis version 7 and later binaries. Webinator 6.2 and later scripts are incompatible with Texis version 6 and earlier binaries. If upgrading binaries to Texis version 7, ensure that any existing scripts are replaced with the included Webinator 6.2 or later scripts (the Texis 7 installer will do this).

E.2 Vortex Features

E.2.1 Script Tracing

In Texis version 7, Vortex script performance tracing tools were added. Execution, with time spent per statement etc., can be logged to a file for later analysis with a GUI tool. This can be used to discover where a script spends most of its execution time, and thus where performance improvement efforts should be focused. See the Trace Vortex appendix (p. 609) for details.
E.2.2 REX Does Not Return Redundant Empty Hits - Caveat

In version 7 and later, REX no longer returns certain theoretically-correct but redundant empty hits. For example, searching for the expression “a*” against the data “a” used to return two hits – “a” and “” (empty string at the end of buffer); in version 7 only the first hit is returned. More specifically, hits are skipped where the entire match (including any \P and \F parts) is empty and adjacent to the previous entire match.

E.2.3 sandr Infinite Loops Prevented, Matches Like rex - Caveat

In version 7 and later, the Vortex <sandr> statement does not infinite-loop on expressions that match an empty string (e.g. “. {,2}”). Also, <sandr> now searches for an expression consistent with how <rex> does. For example, this means that start/end of buffer anchor sub-expressions (e.g. “>=” ) are now respected properly.

E.2.4 sum Action Determined by Format - Caveat

In version 7 and later, the format (first) argument to <sum> determines its behavior – string formats cause concatenation, numeric formats cause addition. In previous versions, the type of the following arguments determined behavior, i.e. numeric addition only happened if all arguments following the format were parsably numeric. This change clarifies <sum> behavior when the non-format arguments are unknown: it is no longer necessary to add a dummy empty string argument to force concatenation.

Caveat: Existing <sum> statements with a string format (%s) but numeric arguments will return different results in version 7 than in version 6.

For example:

`<sum "%s" 1 2 3 4>`

will return the string “10” in version 6: numeric addition first due to numeric arguments, then conversion to string by the format. But in version 7, this will return the string “1234”: the format causes string concatenation. See the Vortex manual on sum (p. 90) for more details. This behavior change is rolled back if <vxcp compatibilityversion> is set to 6.

E.2.5 fmt %l/, %l: Codes Changed - Caveat

In version 7.00.1352409000 20121108 and later, the <fmt> flag and code combinations %l/ and %l: – which print REX expressions to match directory separators and search path separators respectively – have slightly different output. The expressions are always full character classes (i.e. bracketed), and do not have a repetition operator (“=” ) appended. This allows the repetition operator to be changed more easily by the user, by simply appending it when printing.
E.2.6 SQL output=xml Does Not Consume Entities - Caveat

In version 7 and later, Vortex <sql output=xml> (p. 27) no longer converts HTML entities in the data to characters before HTML-escaping the output (for XML). This allows the XML output to represent the same data as the source – even if the latter contains entities – properly and consistently escaped. For example, the SQL column Text with data “Mike &quot;Ike”” would be output as “<Text>Mike &amp; &quot;Ike&quot;</Text>” – the entities are escaped to preserve them, even invalid ones. This behavior change is rolled back if <vxcp compatibilityversion> is set to 6.

E.2.7 Empty-element Statement Tags Allowed - Caveat

In version 7 and later, empty-element ("self-closing" e.g. <element/> ) Vortex statements are permitted. For block or looping statements (e.g. <rex>, <sql> ), this causes the statement to be non-looping; for non-block/non-loop statements (e.g. <sum>), it has no effect. This syntax can be used to "close" Vortex statements lexically to allow SGML-syntx-aware editors to indent and highlight code better.

Caveat: If any existing Vortex statements have a non-quoted trailing forward slash, in Texis version 7 the trailing slash will likely be interpreted as part of a self-closing tag, instead of a separate parameter to the statement. This may cause compile errors and/or missing parameters in Texis version 7. E.g. <case />, while valid in version 6 and earlier as a check for "/", will fail in version 7, as it appears to be missing an argument. The fix is to quote the slash (quoting literal arguments is recommended practice anyway).

E.2.8 Pragma Stacks - Caveat

In Texis version 7 and later, Vortex compiler pragmas (p. 82) can be pushed on and popped off of a stack, with the addition of the push or pop keyword. This allows a pragma to be used locally in a code block, preserving and restoring the previous value – even if unknown.

Also, in version 7 pragmas only affect the <script> block they are in; they are not inherited by following blocks or modules.

E.2.9 write output, flags, skiponfail Options - Caveat

In version 7 and later, the <write> statement (p. 42) also supports the output flag as used with capture, and the flags option was also added.

Caveat: If any existing code writes to a file literal named “output” or “flags”, such filenames may be interpreted as options in version 7 and the code may fail to compile. The fix is simply to move the filename to a variable, or qualify it with a path.

Also in version 7, the skiponfail option was added: if set, the <write> block will be skipped if the file fails to be opened (as in previous versions).
Caveat: skiponfail defaults to off – so <write> statements that fail to open the file will now enter the <write> block and execute the code therein. Previous versions skipped the block on file-open failure.

E.2.10  getvar, setvar Error Messages

In Texit version 7 and later, the Vortex getvar and setvar functions (p. 281 and 283) will issue putmsg messages on certain additional errors: if the variable name is illegal, did not exist at compile time, or is a reserved special variable that cannot be set. These messages allow easier detection of failed calls. The messages may be suppressed with the nomsg flag to getvar or setvar. This behavior change is rolled back if <vxcp compatibilityversion> is set to 6.

E.2.11  vxcp timeouttext and exceptiononly Added

In version 7, the <vxcp timeouttext> and putmsg ... exceptiononly options were added (p. 284). These can be used to better control what is printed when a Vortex script times out, including preventing any message from being output at all, which may be useful if precise XML output is being generated.

E.2.12  vxcp compatibilityversion Added

In version 7, the <vxcp compatibilityversion> option was added (p. 284). This can be used to set compatibility with an earlier version of Texis in legacy scripts, allowing them to run correctly on an upgraded Texis version. There is also a SQL compatibilityversion property, for SQL-only scripts.

E.2.13  Cookie Acceptance Relaxed - Caveat

In version 7, the acceptance of HTTP cookies by the <fetch> library is relaxed and more in line with the latest RFC (6265). Previous releases did not accept cookies with a Path value that did not contain a leading period.

E.2.14  xmlTreeXPathSetContext() Added to XML API

The xmlTreeXPathSetContext() function (p. 480) was added in version 7.

E.2.15  $ret.size is int64 On All Platforms

The Vortex $ret.size variable is type int64 on all platforms in version 7 and later; previously it was long on 32-bit-file platforms.
E.3 SQL Features

E.3.1 strlst/IN Changes and New SUBSET/INTERSECT Operators

In Texis version 7, significant strlst and IN operator behavior was changed, and a few new strlst operators were added. The following is an overview of the changes; see the Texis manual section on IN, SUBSET and INTERSECT for further details and syntax. These changes (except for the new operators SUBSET and INTERSECT) can be rolled back with <vxcp compatibilityversion 6>.

IN behavior change - Caveat

Caveat: The IN operator behaves differently in Texis version 7 and later – it acts more like the new SUBSET operator than INTERSECT. Thus, it is true (matches) if and only if all left-side value(s) are present on the right, not just if any are present (as in version 6 and earlier).

The version 6 behavior can be restored globally with the inmode SQL property (or use the INTERSECT operator); it is also restored when <vxcp compatibilityversion> is set to 6. The IN operator also utilizes indexes more effectively in version 7, which is aided by the new indexvalues SQL property.

New operators SUBSET and INTERSECT - Caveat

The SQL operators SUBSET and INTERSECT were added in version 7, for set-like queries on strlst values. SUBSET is true if and only if all left-side value(s) are present on the right side; INTERSECT is true if any left-side values are present on the right.

SUBSET and INTERSECT are new keywords in Texis version 7. If either is already in use (in all-upper or all-lower case) as an identifier – table name, column name, index name, etc. – then its usage must be enclosed in double-quotes in SQL in version 7, to prevent syntax errors. E.g.: SELECT "subset" FROM MyTable ...

Relational operators with strlst changed - Caveat

The use of relational operators (<, >, = etc.) with strlst values has been clarified somewhat in version 7. While usage with strlst is discouraged – as strlst values are set-like but the relational operators are for scalars – a strlst relational expression will now generally compare the two strlsts as atomic entities, promoting the non-strlst side if present (as a single item). Version 6 behavior is restored if <vxcp compatibilityversion> is set to 6.

stringcomparemode used with strlst - Caveat

In version 7, the current stringcomparemode setting is used when comparing strlst values or individual items. This means case-insensitive set (and relational) operations can be performed, by altering stringcomparemode. Version 6 behavior (ignore stringcomparemode) is restored if <vxcp compatibilityversion> is set to 6.
APPENDIX E. TEXIS VERSION 7 FEATURES AND CHANGES

varchartostrlstsep default is now create - Caveat

In version 7, the default value for the varchartostrlstsep SQL property changed to create (it was previously lastchar). This simplifies the passing of Vortex string arrays to SQL strlst columns, especially when they may occasionally contain a single value.

Caveat: As a consequence, a single string such as “a,b,c,” that was previously split into three values upon conversion to strlst, will be a single-value strlst (“a,b,c,”) in version 7.

There are several ways to restore the version 6 default: set the [Texis] Varchar To Strlst Sep setting in conf/texis.ini; set the varchartostrlstsep SQL property; use the SQL convert() function with a third argument of “lastchar”; or set <vxcp compatibilityversion> to 6. See the Texis manual on these settings and the convert() function for details.

Duplicate multi-item strlst index results removed - Caveat

In version 7, duplicate result rows caused by strlst indexes (with indexvalues set to split, the default) are removed, so that results are more consistent with linear searches: a given row should be represented at most once in the results.

Caveat: This means that many ORDER BY, relational etc. queries that use strlst indexes will have a lower total result count. The same rows will be returned, just without duplicates.

This behavior change is rolled back if <vxcp compatibilityversion> is set to 6.

multivaluetomultirow defaults off - Caveat

In version 7, the multivaluetomultirow SQL property defaults to off, e.g. strlst values are not split into items first by GROUP BY and DISTINCT. This keeps consistency with the removal of duplicate strlst index results: a given row should generally be represented only once in the results. To revert behavior, set the [Texis] Multi Value To Multi Row setting in texis.ini to on, or set the multivaluetomultirow SQL property to on, or set <vxcp compatibilityversion> to 6.

E.3.2 ALTER INDEX Syntax Added

In Texis version 7, Metamorph indexes are easier to update by using the new ALTER INDEX syntax, which avoids the need to specify the complete table and fields. For example:

    ALTER INDEX searchIndex OPTIMIZE;

is equivalent to:

    CREATE METAMORPH INDEX searchIndex
        ON searchTable(Title\Meta\Body, Price, Count);
E.3. SQL FEATURES

(assuming searchIndex already exists). Search the online Texis manual for "ALTER INDEX" for details.

E.3.3 CREATE INDEX ... WITH options Syntax - Caveat

In Texis version 7, many index options that were previously only alterable via global SQL properties may be specified directly in the SQL CREATE INDEX statement, e.g. index expressions:

```
CREATE METAMORPH INVERTED INDEX searchIndex
ON myTable(Title\Meta\Body,Price)
WITH WORDEXPRESSIONS ('\alnum{2,99}', '[\x80-\xFF]{2,99}')
    KEEPNOISE 'on' INDEXMETER 'on';
```

This helps reduce side-effects that can occur when a global setting was changed to create one index but erroneously left on for another: the setting can now be moved into the creation statement itself. Search the online Texis manual for "Index Options" for details.

The keyword FULLTEXT was added in Texis version 7 as a clearer synonym for METAMORPH. If FULLTEXT (all-upper or all-lower) is already in use as an identifier – table name, column name, index name, etc. – then its usage must be enclosed in double-quotes in SQL in version 7, to prevent syntax errors. E.g.: SELECT "fulltext" FROM MyTable ...

E.3.4 Metamorph Index Default Type Is INVERTED - Caveat

In Texis version 7 and later, the default Metamorph index type is INVERTED, as it is more useful than non-inverted and the most commonly used variant.

Caveat: Thus, in version 7 CREATE METAMORPH INDEX ... (with no other options) will create an inverted Metamorph index, not a non-inverted Metamorph as in previous versions.

To create a non-inverted index in version 7 and later, use the WORDPOSITIONS 'off' index option. Search the online Texis manual for "Index Options" for more details. This behavior change is rolled back if <vxcp compatibilityversion> is set to 6.

E.3.5 byte/char Conversion As-is Without Hexifying - Caveat

Caveat: In version 7 and later, converting byte values to char (and vice versa) copies the data as-is (stopping at a null byte), without converting to (or from) hexadecimal.

This makes it easier to manipulate and view varbyte table columns that may contain polymorphic but often-textual data, and maintains consistency with Vortex, which never implicitly converts to or from hexadecimal when dealing with byte data. To convert data to hex or back in version 7, use the new hextobin() and bintohex() SQL functions. This behavior change is rolled back if <vxcp compatibilityversion> is set to 6.