Copyright

Copyright © 2001 BEA Systems, Inc. All Rights Reserved.

Restricted Rights Legend

This software and documentation is subject to and made available only pursuant to the terms of the BEA Systems License Agreement and may be used or copied only in accordance with the terms of that agreement. It is against the law to copy the software except as specifically allowed in the agreement. This document may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from BEA Systems, Inc.

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the BEA Systems License Agreement and in subparagraph (c)(1) of the Commercial Computer Software-Restricted Rights Clause at FAR 52.227-19; subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, subparagraph (d) of the Commercial Computer Software--Licensing clause at NASA FAR supplement 16-52.227-86; or their equivalent.

Information in this document is subject to change without notice and does not represent a commitment on the part of BEA Systems. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER, BEA Systems DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF THE USE, OF THE SOFTWARE OR WRITTEN MATERIAL IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

Trademarks or Service Marks


All other product names may be trademarks of the respective companies with which they are associated.

ATMI FML Function Reference

<table>
<thead>
<tr>
<th>Document Edition</th>
<th>Date</th>
<th>Software Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.0</td>
<td>June 2001</td>
<td>BEA Tuxedo Release 8.0</td>
</tr>
</tbody>
</table>
Contents

About This Document

Section 3fml - FML Functions
Introduction to FML Functions .................................................................6
CFadd, CFadd32(3fml) ........................................................................12
CFchg, CFchg32(3fml) ......................................................................14
CFfind, CFfind32(3fml) ...................................................................16
CFfindocc, CFfindocc32(3fml) ......................................................18
CFget, CFget32(3fml) .....................................................................20
CFgetalloc, CFgetalloc32(3fml) ....................................................22
F_error, F_error32(3fml) ................................................................24
F32to16, F16to32(3fml) ...............................................................25
Fadd, Fadd32(3fml) ........................................................................27
Fadds, Fadds32(3fml) .....................................................................29
Falloc, Falloc32(3fml) ....................................................................31
Fappend, Fappend32(3fml) .............................................................32
Fboolco, Fboolco32, Fvboolco, Fvboolco32(3fml) .....................34
Fboolev, Fboolev32, Fvboolev, Fvboolev32(3fml) .......................37
Fboolpr, Fboolpr32, Fvboolpr, Fvboolpr32(3fml) ....................40
Fchg, Fchg32(3fml) ......................................................................42
Fchgs, Fchgs32(3fml) ...................................................................44
Fchksum, Fchksum32(3fml) ..........................................................46
Fcmp, Fcmp32(3fml) .....................................................................47
Fconcat, Fconcat32(3fml) ..............................................................49
Fcpy, Fcpy32(3fml) .......................................................................50
Fdel, Fdel32(3fml) ........................................................................51
Fdelall, Fdelall32(3fml) ................................................................53
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fdelete, Fdelete32(3fml)</td>
<td>55</td>
</tr>
<tr>
<td>Fextread, Fextread32(3fml)</td>
<td>57</td>
</tr>
<tr>
<td>Ffind, Ffind32(3fml)</td>
<td>60</td>
</tr>
<tr>
<td>Ffindlast, Ffindlast32(3fml)</td>
<td>62</td>
</tr>
<tr>
<td>Ffindoce, Ffindoce32(3fml)</td>
<td>64</td>
</tr>
<tr>
<td>Ffinds, Ffinds32(3fml)</td>
<td>66</td>
</tr>
<tr>
<td>Ffloatev, Ffloatev32, Fvfloatev, Fvfloatev32(3fml)</td>
<td>68</td>
</tr>
<tr>
<td>Ffprint, Ffprint32(3fml)</td>
<td>70</td>
</tr>
<tr>
<td>Ffree, Ffree32(3fml)</td>
<td>72</td>
</tr>
<tr>
<td>Fget, Fget32(3fml)</td>
<td>73</td>
</tr>
<tr>
<td>Fgetalloc, Fgetalloc32(3fml)</td>
<td>75</td>
</tr>
<tr>
<td>Fgetlast, Fgetlast32(3fml)</td>
<td>77</td>
</tr>
<tr>
<td>Fgets, Fgets32(3fml)</td>
<td>79</td>
</tr>
<tr>
<td>Fgetsa, Fgetsa32(3fml)</td>
<td>81</td>
</tr>
<tr>
<td>Fidnm_unload, Fidnm_unload32(3fml)</td>
<td>83</td>
</tr>
<tr>
<td>Fidxused, Fidxused32(3fml)</td>
<td>84</td>
</tr>
<tr>
<td>Fielded, Fielded32(3fml)</td>
<td>85</td>
</tr>
<tr>
<td>Findex, Findex32(3fml)</td>
<td>86</td>
</tr>
<tr>
<td>Finit, Finit32(3fml)</td>
<td>87</td>
</tr>
<tr>
<td>Fjoin, Fjoin32(3fml)</td>
<td>88</td>
</tr>
<tr>
<td>Fldid, Fldid32(3fml)</td>
<td>90</td>
</tr>
<tr>
<td>Fldno, Fldno32(3fml)</td>
<td>91</td>
</tr>
<tr>
<td>Fldtype, Fldtype32(3fml)</td>
<td>92</td>
</tr>
<tr>
<td>Flen, Flen32(3fml)</td>
<td>93</td>
</tr>
<tr>
<td>Fmkfldid, Fmkfldid32(3fml)</td>
<td>95</td>
</tr>
<tr>
<td>Fmove, Fmove32(3fml)</td>
<td>96</td>
</tr>
<tr>
<td>Fname, Fname32(3fml)</td>
<td>98</td>
</tr>
<tr>
<td>Fneeded, Fneeded32(3fml)</td>
<td>99</td>
</tr>
<tr>
<td>Fnext, Fnext32(3fml)</td>
<td>100</td>
</tr>
<tr>
<td>Fnmid_unload, Fnmid_unload32(3fml)</td>
<td>102</td>
</tr>
<tr>
<td>Fnum, Fnum32(3fml)</td>
<td>103</td>
</tr>
<tr>
<td>Foccur, Foccur32(3fml)</td>
<td>104</td>
</tr>
<tr>
<td>Fojoin, Fojoin32(3fml)</td>
<td>105</td>
</tr>
<tr>
<td>Fpres, Fpres32(3fml)</td>
<td>107</td>
</tr>
<tr>
<td>Fprint, Fprint32(3fml)</td>
<td>108</td>
</tr>
</tbody>
</table>
Fproj, Fproj32(3fml).................................................................110
Fprojcpy, Fprojcpy32(3fml)..................................................112
Fread, Fread32(3fml)..........................................................114
Frealloc, Frealloc32(3fml)...............................................116
Frstrindex, Frstrindex32(3fml)......................................118
Fsizeof, Fsizeof32(3fml) ..................................................120
Fstrerror, Fstrerror32(3fml).........................................121
Ftypecv, Ftypecv32(3fml) ..............................................122
Ftype, Ftype32(3fml)..........................................................124
Funindex, Funindex32(3fml)........................................125
Funused, Funused32(3fml)............................................127
Fupdate, Fupdate32(3fml)..............................................128
Fused, Fused32(3fml)......................................................130
Fvall, Fvall32(3fml)........................................................131
Fvals, Fvals32(3fml)......................................................133
Fvftos, Fvftos32(3fml)..................................................135
Fvneeded, Fvneeded32(3fml)....................................137
Fvnull, Fvnull32(3fml)................................................138
Fvopt, Fvopt32(3fml)...................................................140
Fvrefresh, Fvrefresh32(3fml)..................................142
Fvselinit, Fvselinit32(3fml)....................................143
Fvsinit, Fvsinit32(3fml).............................................145
Fvstof, Fvstof32(3fml)..............................................146
Fvstot, Fvstos(3fml)...................................................148
Fwrite, Fwrite32(3fml)................................................154
About This Document

This document provides reference information on Field Manipulation Language (FML) functions used in the BEA Tuxedo ATMI environment. FML is a set of C functions for defining and manipulating fielded buffers. The reference pages are arranged in alphabetical order by function name.

What You Need to Know

This document is intended for the following audiences:

- administrators who are interested in configuring and managing applications in a BEA Tuxedo environment
- application developers who are interested in programming applications in a BEA Tuxedo environment

This document assumes a familiarity with the BEA Tuxedo platform and C programming.

e-docs Web Site

BEA product documentation is available on the BEA corporate Web site. From the BEA Home page, click on Product Documentation or go directly to the “e-docs” Product Documentation page at http://e-docs.bea.com.
How to Print the Document

You can print a copy of this document from a Web browser, one file at a time, by using the File—>Print option on your Web browser.

A PDF version of this document is available on the BEA Tuxedo documentation Home page on the e-docs Web site (and also on the documentation CD). You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format. To access the PDFs, open the BEA Tuxedo documentation Home page, click the PDF files button and select the document you want to print.

If you do not have the Adobe Acrobat Reader, you can get it for free from the Adobe Web site at http://www.adobe.com/.

Related Information

Related documents are listed in the See Also section of each reference page.

Contact Us!

Your feedback on the BEA Tuxedo documentation is important to us. Send us e-mail at docsupport@bea.com if you have questions or comments. Your comments will be reviewed directly by the BEA professionals who create and update the BEA Tuxedo documentation.

In your e-mail message, please indicate that you are using the documentation for the BEA Tuxedo 8.0 release.
If you have any questions about this version of BEA Tuxedo, or if you have problems installing and running BEA Tuxedo, contact BEA Customer Support through BEA WebSupport at www.bea.com. You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

When contacting Customer Support, be prepared to provide the following information:

- Your name, e-mail address, phone number, and fax number
- Your company name and company address
- Your machine type and authorization codes
- The name and version of the product you are using
- A description of the problem and the content of pertinent error messages

**Documentation Conventions**

The following documentation conventions are used throughout this document.

<table>
<thead>
<tr>
<th>Convention</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>boldface text</strong></td>
<td>Indicates terms defined in the glossary.</td>
</tr>
<tr>
<td>Ctrl+Tab</td>
<td>Indicates that you must press two or more keys simultaneously.</td>
</tr>
<tr>
<td><em>italics</em></td>
<td>Indicates emphasis or book titles.</td>
</tr>
</tbody>
</table>
### Convention Item

<table>
<thead>
<tr>
<th>Convention</th>
<th>Item</th>
</tr>
</thead>
</table>
| **monospace text** | Indicates code samples, commands and their options, data structures and their members, data types, directories, and file names and their extensions. Monospaced text also indicates text that you must enter from the keyboard.  
*Examples:*  
\`
#include <iostream.h> void main () the pointer psz
chmod u+w *
\tux\data\ap
.doc
tux.doc
BITMAP
float
\`
| **monospace boldface text** | Identifies significant words in code.  
*Example:*  
\`
void commit ()
\`
| **monospace italic text** | Identifies variables in code.  
*Example:*  
\`
String expr
\`
| **UPPERCASE TEXT** | Indicates device names, environment variables, and logical operators.  
*Examples:*  
\`
LPT1
SIGNON
OR
\`
| **{}** | Indicates a set of choices in a syntax line. The braces themselves should never be typed.  
*Example:*  
\`
builde\bject\cj\ient \[-v\] \[-o name \] \[-f file-list]...  
\[-l file-list]\...
\`
| **[]** | Indicates optional items in a syntax line. The brackets themselves should never be typed.  
*Example:*  
\`
builde\bject\cj\ient \[-v\] \[-o name \] \[-f file-list]...  
\[-l file-list]\...
\`
| **|** | Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.  
*Example:*  
\`
builde\bject\cj\ient \[-v\] \[-o name \] \[-f file-list]...  
\[-l file-list]\...
\`

---

ATMI FML Function Reference
<table>
<thead>
<tr>
<th>Convention</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>Indicates one of the following in a command line:</td>
</tr>
<tr>
<td></td>
<td>- That an argument can be repeated several times in a command line</td>
</tr>
<tr>
<td></td>
<td>- That the statement omits additional optional arguments</td>
</tr>
<tr>
<td></td>
<td>- That you can enter additional parameters, values, or other information</td>
</tr>
<tr>
<td></td>
<td>The ellipsis itself should never be typed.</td>
</tr>
<tr>
<td></td>
<td><em>Example:</em></td>
</tr>
<tr>
<td></td>
<td><code>buildobjclient [-v] [o name ] [o file-list]... [l file-list]...</code></td>
</tr>
<tr>
<td>.</td>
<td>Indicates the omission of items from a code example or from a syntax line.</td>
</tr>
<tr>
<td>.</td>
<td>The vertical ellipsis itself should never be typed.</td>
</tr>
</tbody>
</table>
### Section 3fml - FML Functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction to FML Functions</td>
<td>Provides an introduction to the FML functions</td>
</tr>
<tr>
<td>CFadd, CFadd32(3fml)</td>
<td>Converts and adds field</td>
</tr>
<tr>
<td>CFchg, CFchg32(3fml)</td>
<td>Converts and changes field</td>
</tr>
<tr>
<td>CFfind, CFfind32(3fml)</td>
<td>Finds, converts, and returns pointer</td>
</tr>
<tr>
<td>CFfindocc, CFfindocc32(3fml)</td>
<td>Finds occurrence of converted value</td>
</tr>
<tr>
<td>CFget, CFget32(3fml)</td>
<td>Gets field and converts</td>
</tr>
<tr>
<td>CFgetalloc, CFgetalloc32(3fml)</td>
<td>Gets field, allocates space, and converts</td>
</tr>
<tr>
<td>F_error, F_error32(3fml)</td>
<td>Prints error message for last error</td>
</tr>
<tr>
<td>F32to16, F16to32(3fml)</td>
<td>Converts 16-bit FML to/from 32-bit FML buffer</td>
</tr>
<tr>
<td>Fadd, Fadd32(3fml)</td>
<td>Adds new field occurrence</td>
</tr>
<tr>
<td>Fadds, Fadds32(3fml)</td>
<td>Converts value from type FLD_STRING and adds to buffer</td>
</tr>
<tr>
<td>Fallocl, Fallocl32(3fml)</td>
<td>Allocates and initializes fielded buffer</td>
</tr>
<tr>
<td>Fappend, Fappend32(3fml)</td>
<td>Appends new field occurrence</td>
</tr>
</tbody>
</table>
### Table 1  BEA Tuxedo ATMI FML Functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fboolco, Fboolco32, Fvboolco, Fvboolco32(3fml)</td>
<td>Compiles expression, returns evaluation tree</td>
</tr>
<tr>
<td>Fboolev, Fboolev32, Fvboolev, Fvboolev32(3fml)</td>
<td>Evaluates buffer against tree</td>
</tr>
<tr>
<td>Fboolpr, Fboolpr32, Fvboolpr, Fvboolpr32(3fml)</td>
<td>Prints Boolean expression as parsed</td>
</tr>
<tr>
<td>Fchg, Fchg32(3fml)</td>
<td>Changes field occurrence value</td>
</tr>
<tr>
<td>Fchgs, Fchgs32(3fml)</td>
<td>Changes field occurrence - caller presents string</td>
</tr>
<tr>
<td>Fchksum, Fchksum32(3fml)</td>
<td>Computes checksum for fielded buffer</td>
</tr>
<tr>
<td>Fcmp, Fcmp32(3fml)</td>
<td>Compares two fielded buffers</td>
</tr>
<tr>
<td>Fconcat, Fconcat32(3fml)</td>
<td>Concatenates source to destination buffer</td>
</tr>
<tr>
<td>Fcpy, Fcpy32(3fml)</td>
<td>Copies source to destination buffer</td>
</tr>
<tr>
<td>Fdel, Fdel32(3fml)</td>
<td>Deletes field occurrence from buffer</td>
</tr>
<tr>
<td>Fdelall, Fdelall32(3fml)</td>
<td>Deletes all field occurrences from buffer</td>
</tr>
<tr>
<td>Fdelete, Fdelete32(3fml)</td>
<td>Deletes list of fields from buffer</td>
</tr>
<tr>
<td>Fextread, Fextread32(3fml)</td>
<td>Builds fielded buffer from printed format</td>
</tr>
<tr>
<td>Ffind, Ffind32(3fml)</td>
<td>Finds field occurrence in buffer</td>
</tr>
<tr>
<td>Ffindlast, Ffindlast32(3fml)</td>
<td>Finds last occurrence of field in buffer</td>
</tr>
<tr>
<td>Ffindocc, Ffindocc32(3fml)</td>
<td>Finds occurrence of field value</td>
</tr>
<tr>
<td>Ffinds, Ffinds32(3fml)</td>
<td>Returns ptr to string representation</td>
</tr>
<tr>
<td>Ffroatev, Ffroatev32, Fvfroatev, Fvfroatev32(3fml)</td>
<td>Returns value of expression as a double</td>
</tr>
<tr>
<td>Ffprint, Ffprint32(3fml)</td>
<td>Prints fielded buffer to specified stream</td>
</tr>
<tr>
<td>Ffree, Ffree32(3fml)</td>
<td>Frees space allocated for fielded buffer</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>Fget, Fget32(3fml)</td>
<td>Gets copy and length of field occurrence</td>
</tr>
<tr>
<td>Fgetalloc, Fgetalloc32(3fml)</td>
<td>Allocates space and gets copy of field occurrence</td>
</tr>
<tr>
<td>Fgetlast, Fgetlast32(3fml)</td>
<td>Gets copy of last occurrence</td>
</tr>
<tr>
<td>Fgets, Fgets32(3fml)</td>
<td>Gets value converted to string</td>
</tr>
<tr>
<td>Fgetsa, Fgetsa32(3fml)</td>
<td>Uses malloc() to allocate space and gets converted value</td>
</tr>
<tr>
<td>Fidnm_unload,</td>
<td>Recovers space from id-&gt;nm mapping tables</td>
</tr>
<tr>
<td>Fidnm_unload32(3fml)</td>
<td></td>
</tr>
<tr>
<td>Fidxused, Fidxused32(3fml)</td>
<td>Returns amount of space used</td>
</tr>
<tr>
<td>Fielded, Fielded32(3fml)</td>
<td>Returns true if buffer is fielded</td>
</tr>
<tr>
<td>Findex, Findex32(3fml)</td>
<td>Indexes a fielded buffer</td>
</tr>
<tr>
<td>Finit, Finit32(3fml)</td>
<td>Initializes fielded buffer</td>
</tr>
<tr>
<td>Fjoin, Fjoin32(3fml)</td>
<td>Joins source into destination buffer</td>
</tr>
<tr>
<td>Fldid, Fldid32(3fml)</td>
<td>Maps field name to field identifier</td>
</tr>
<tr>
<td>Fldno, Fldno32(3fml)</td>
<td>Maps field identifier to field number</td>
</tr>
<tr>
<td>Fldtype, Fldtype32(3fml)</td>
<td>Maps field identifier to field type</td>
</tr>
<tr>
<td>Flen, Flen32(3fml)</td>
<td>Returns len of field occurrence in buffer</td>
</tr>
<tr>
<td>Fmkfldid, Fmkfldid32(3fml)</td>
<td>Makes a field identifier</td>
</tr>
<tr>
<td>Fmove, Fmove32(3fml)</td>
<td>Moves fielded buffer to destination</td>
</tr>
<tr>
<td>Fname, Fname32(3fml)</td>
<td>Maps field identifier to field name</td>
</tr>
<tr>
<td>Fneeded, Fneeded32(3fml)</td>
<td>Computes size needed for buffer</td>
</tr>
<tr>
<td>Fnext, Fnext32(3fml)</td>
<td>Gets next field occurrence</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>Fnmid_unload,</td>
<td>Recovers space from nm-&gt;id mapping tables</td>
</tr>
<tr>
<td>Fnmid_unload32(3fml)</td>
<td></td>
</tr>
<tr>
<td>Fnum, Fnum32(3fml)</td>
<td>Returns count of all occurrences in buffer</td>
</tr>
<tr>
<td>Foccur, Foccur32(3fml)</td>
<td>Returns count of field occurrences in buffer</td>
</tr>
<tr>
<td>Fojoin, Fojoin32(3fml)</td>
<td>Outer join of source into destination buffer</td>
</tr>
<tr>
<td>Fpres, Fpres32(3fml)</td>
<td>True if field occurrence is present in buffer</td>
</tr>
<tr>
<td>Fprint, Fprint32(3fml)</td>
<td>Prints buffer to standard output</td>
</tr>
<tr>
<td>Fproj, Fproj32(3fml)</td>
<td>Provides projection on buffer</td>
</tr>
<tr>
<td>Fprojcpy, Fprojcpy32(3fml)</td>
<td>Provides projection and copy on buffer</td>
</tr>
<tr>
<td>Fread, Fread32(3fml)</td>
<td>Reads fielded buffer</td>
</tr>
<tr>
<td>Frealloc, Frealloc32(3fml)</td>
<td>Reallocates fielded buffer</td>
</tr>
<tr>
<td>Frstrindex, Frstrindex32(3fml)</td>
<td>Restores index in a buffer</td>
</tr>
<tr>
<td>Fsizeof, Fsizeof32(3fml)</td>
<td>Returns size of fielded buffer</td>
</tr>
<tr>
<td>Fstrerror, Fstrerror32(3fml)</td>
<td>Gets error message string for FML error</td>
</tr>
<tr>
<td>Ftypcvt, Ftypcvt32(3fml)</td>
<td>Converts from one field type to another</td>
</tr>
<tr>
<td>Ftype, Ftype32(3fml)</td>
<td>Returns pointer to type of field</td>
</tr>
<tr>
<td>Funindex, Funindex32(3fml)</td>
<td>Discards fielded buffer's index</td>
</tr>
<tr>
<td>Funused, Funused32(3fml)</td>
<td>Returns number of unused bytes in fielded buffer</td>
</tr>
<tr>
<td>Fupdate, Fupdate32(3fml)</td>
<td>Updates destination buffer with source</td>
</tr>
<tr>
<td>Fused, Fused32(3fml)</td>
<td>Returns number of used bytes in fielded buffer</td>
</tr>
<tr>
<td>Fvall, Fvall32(3fml)</td>
<td>Returns long value of field occurrence</td>
</tr>
<tr>
<td>Fvals, Fvals32(3fml)</td>
<td>Returns string value of field occurrence</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>Fvftos, Fvftos32(3fml)</td>
<td>Copies from fielded buffer to C structure</td>
</tr>
<tr>
<td>Fvneeded, Fvneeded32(3fml)</td>
<td>Computes size needed for view buffer</td>
</tr>
<tr>
<td>Fvnull, Fvnull32(3fml)</td>
<td>Checks if a structure element is NULL</td>
</tr>
<tr>
<td>Fvopt, Fvopt32(3fml)</td>
<td>Changes flag options of a mapping entry</td>
</tr>
<tr>
<td>Fvrefresh, Fvrefresh32(3fml)</td>
<td>Copies from C structure to fielded buffer</td>
</tr>
<tr>
<td>Fvselinit, Fvselinit32(3fml)</td>
<td>Initializes structure element to NULL</td>
</tr>
<tr>
<td>Fvsinit, Fvsinit32(3fml)</td>
<td>Initializes C structure to NULL</td>
</tr>
<tr>
<td>Fvstof, Fvstof32(3fml)</td>
<td>Copies from C structure to fielded buffer</td>
</tr>
<tr>
<td>Fvstot, Fvttos(3fml)</td>
<td>Converts C structure to/from target record type</td>
</tr>
<tr>
<td>Fwrite, Fwrite32(3fml)</td>
<td>Writes fielded buffer</td>
</tr>
</tbody>
</table>
Introduction to FML Functions

Synopsis

```c
#include <fml.h>
#include <fml32.h>
```

Description

FML is a set of C language functions for defining and manipulating storage structures called fielded buffers, that contain attribute-value pairs called fields. The attribute is the field’s identifier, and the associated value represents the field’s data content.

Fielded buffers provide an excellent structure for communicating parameterized data between cooperating processes, by providing named access to a set of related fields. Programs that need to communicate with other processes can use the FML software to provide access to fields without concerning themselves with the structures containing them.

FML also provides a facility called views that allows you to map fielded buffers to C structures (and the reverse as well). Views lets you perform lengthy manipulations of data in structures rather than in fielded buffers; applications will run faster if data is transferred to structures for manipulation. Views allows the data independence of fielded buffers to be combined with the efficiency and simplicity of classic record structures.

FML16 and FML32

There are two “sizes” of FML. The original FML interface is based on 16-bit values for the length of fields and containing information identifying fields. In this introduction, it will be referred to as FML16. FML16 is limited to 8191 unique fields, individual field lengths of up to 64K bytes, and a total fielded buffer size of 64K. The definitions, types, and function prototypes for this interface are in fml.h which must be included in an application program using the FML16 interface; and functions live in -1fml. A second interface, FML32, uses 32-bit values for the field lengths and identifiers. It allows for about 30 million fields, and field and buffer lengths of about 2 billion bytes. The definitions, types, and function prototypes for FML32 are in fml32.h; and functions live in -1fml32. All definitions, types, and function names for FML32 have a “32” suffix (for example, MAXFLEN32, FLDID32, Fchg32). Also the environment variables are suffixed with “32” (for example, FLDTBLDIR32, FIELDTBLS32, VIEWFILES32, and VIEWDIR32).

FML Buffers

A fielded buffer is composed of field identifier and field value pairs for fixed length fields (for example, long, short), and field identifier, field length, and field value triples for varying length fields.
A field identifier is a tag for an individual data item in a fielded buffer. The field identifier consists of the name of field number and the type of the data in the field. The field number must be in the range 1 to 8191 inclusive for FML16, and the type definition for a field identifier is FLDID. The field number must be in the range 1 to 33,554,431 inclusive for FML32, and the type definition for a field identifier is FLDID32. The BEA Tuxedo ATMI system conforms to the following conventions for field numbers:

<table>
<thead>
<tr>
<th>FML16 Field Numbers</th>
<th>FML32 Field Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Available</td>
</tr>
<tr>
<td>1-100</td>
<td>101-8191</td>
</tr>
<tr>
<td></td>
<td>1-10,000,</td>
</tr>
<tr>
<td></td>
<td>30,000,001-33,554,431</td>
</tr>
</tbody>
</table>

Applications should avoid using the reserved field numbers, although the BEA Tuxedo ATMI system does not strictly enforce applications from using them.

The field types can be any of the standard C language types: short, long, float, double, and char. The following types are also supported: string (a series of characters ending with a NULL character), carray (character arrays), pointer (a pointer to a buffer), FML32 buffer (an embedded FML32 buffer), and VIEW32 (an embedded VIEW32 buffer). These types are defined in fml.h and fml32.h as FLD_SHORT, FLD_LONG, FLD_CHAR, FLD_FLOAT, FLD_DOUBLE, FLD_STRING, FLD_CARRAY, FLD_PTR, FLDFML32, and FLD_VIEW32.

For FML16, a fielded buffer pointer is of type FBFR *, a field length has the type FLDLEN, and the number of occurrences of a field has the type FLDLOC. For FML32, a fielded buffer pointer is of type FBFR32 *, a field length has the type FLDLEN32, and the number of occurrences of a field has the type FLDLOC32.

Fields are referred to by their field identifier in the FML interface. However, it is normally easier for an application programmer to remember a field name. There are two approaches to mapping field names to field identifiers.

Field name/identifier mappings can be made available to FML programs at run time through field table files, described in field_tables(5). The FML16 interface uses the environment variable FLDTBLDIR to specify a list of directories where field tables can be found, and FIELDTBLS to specify a list of the files in the table directories that are to be used. The FML32 interface uses FLDTBLDIR32 and FIELDTBLS32. Within
applications programs, the FML functions Flid() and Flid32() provide for a run-time translation of a field name to its field identifier and Fname() and Fname32() translate a field identifier to its field name.

Compile-time field name/identifier mappings are provided by the use field header files containing macro definitions for the field names. mkfldhdr() and mkfldhdr32() are provided to make header files out of field table files (see mkfldhdr, mkfldhdr32(1) for details). These header files are #include'd in C programs, and provide another way to map field names to field identifiers at compile-time.

Any field in a fielded buffer can occur more than once. Many FML functions take an argument that specifies which occurrence of a field is to be retrieved or modified. If a field occurs more than once, the first occurrence is numbered 0, and additional occurrences are numbered sequentially. The set of all occurrences make up a logical sequence, but no overhead is associated with the occurrence number (that is, it is not stored in the fielded buffer). If another occurrence of a field is added, it is added at the end of the set and is referred to as the next higher occurrence. When an occurrence other than the highest is deleted, all higher occurrences of the field are shifted down by one (for example, occurrence 6 becomes occurrence 5, 5 becomes 4, etc.).

When a fielded buffer has many fields, access is expedited in FML by the use of an internal index. The user is normally unaware of the existence of this index. However, when you store a fielded buffer on disk, or transmit a fielded buffer between processes or between computers, you can save disk space and/or transmittal time by first discarding the index using Funindex() or Funindex32(), and then reconstructing the index later with Findex() or Findex32().
performance. Information in a fielded buffer can be extracted from the fields in a buffer and placed in a C structure using VIEWS functions, manipulated, and the updated values returned to the buffer, again using VIEWS functions.

Typed buffers is a feature of the ATMI environment that grew out of the FML idea of a fielded buffer. Two of the standard buffer types delivered with the ATMI environment are FML typed buffers and VIEW typed buffers. An additional difference of VIEW buffers is that they can be totally unrelated to an FML fielded buffer. The buffer types FML32 and VIEW32 can also be used.

A view description is created and stored in a source viewfile, as described in viewfile(5). The view description maps fields in fielded buffers to members in C structures. The source view descriptions are compiled, using viewc() or viewc32(), creating a view object file and can then be used to map data transferred between fielded buffers and C structures in a C program (see viewc, viewc32(1) for details). The view compiler also creates C header files that can be included in applications programs to define the structures described in view descriptions. A view disassembler, viewdis() or viewdis32(), is provided to translate object view descriptions into readable form (that is, back into source view descriptions); the output of the disassembler can be reinput to the view compiler (see viewdis, viewdis32(1) for details).

The object files are used at run time to manipulate the VIEW structures using the VIEWFILES and VIEWDIR environment variables. VIEWFILES should contain a comma-separated list of object viewfiles for the application. Files given as full pathnames are used as is; files listed as relative pathnames are searched for through the list of directories specified by the VIEWDIR variable (as described later in this section). VIEWDIR specifies a colon-separated list of directories to be used to find view object files with relative filenames. For VIEW32 structures, VIEWFILES32 and VIEWDIR32 are used.

In addition to the data types supported by most FML functions, VIEWS supports type int in source view descriptions. When the view description is compiled the view compiler automatically converts any int types to either short or long types, depending on your machine.

A decimal data type is also supported in VIEWS. It is defined as a field of type dec_t, and the size of the packed decimal value is given as the total number of bytes and the bytes to the right of the decimal point. While this field is not supported directly in FML, conversion of this field is automatic to/from any other field type supported in FML. Packed decimals exist in the COBOL environment as two decimal digits packed
Section 3fml - FML Functions

into one byte with the low-order half byte used to store the sign. In the C environment, the data type is defined by the `dec_t` type definition, which contains the decimal exponent, sign, digits, and the packed decimal value.

An FML buffer can be converted to a view using `Fvftos()` or `Fvftos32()`. A view can be converted to a fielded buffer using `Fvstof()` or `Fvstof32()`. When transferring data between fielded buffers and structures, the source data is automatically converted to the type of the destination data. Multiple field occurrences are supported; they are treated as an array in the structure. NULL values are used to indicate empty members in a structure, and can be specified by the user for each structure member in a viewfile. If the user does not specify a NULL value for a member, default NULL values are used. It is also possible to inhibit the transfer of data between a C structure member and a field in a fielded buffer, even though a mapping exists between them.

A view can also be converted to and from a target record format. The default target format is IBM System/370 COBOL records. The `Fvstof()` function takes care of converting byte ordering, floating point and decimal format, and character sets (ASCII to EBCDIC), and `Fvftos()` converts back to the native format. 32-bit versions of these functions also exist. The `Fcodeset()` function can be used to specify alternate ASCII/EBCDIC transaction tables.

**Error Handling**

Most of the FML functions have one or more error returns. An error condition is indicated by an otherwise impossible returned value. This is usually -1 on error, or 0 for a bad field identifier (BADFLDID) or address. The error type is also made available in the external integer `Ferror` for FML16 and `Ferror32` for FML32. `Ferror` and `Ferror32` are not cleared on successful calls, so they should be tested only after an error has been indicated.

The `F_error` and `F_error32` functions are provided to produce a message on the standard error output. They take one parameter, a string; print the argument string appended with a colon and a blank; and then print an error message followed by a newline character. The error message displayed is the one defined for the error number currently in `Ferror` or `Ferror32`, which is set when errors occur.

`Fstrerror()` can be used to retrieve from a message catalog the text of an error message; it returns a pointer that can be used to as an argument to `userlog(3c)`.

The error codes that can be produced by an FML function are described on each FML reference page.
Introduction to FML Functions

See Also
- CFadd, CFadd32(3fml), CFchg, CFchg32(3fml), CFfind, CFfind32(3fml), CFfindocc, CFfindocc32(3fml), CFget, CFget32(3fml), CFgetalloc, CFgetalloc32(3fml), F_error, F_error32(3fml), Fadd, Fadd32(3fml)
- Fadds, Fadds32(3fml), Falloc, Falloc32(3fml), Fboolco, Fboolco32, Fboolev, Fboolev32, Fboolpr, Fboolpr32
- Fboolco, Fboolco32, Fvboolco, Fvboolco32(3fml), Fvboolev, Fvboolev32, Fvboolpr, Fvboolpr32
- F BOOLCO, FBOOLCO32(3FML), Fcopy, Fcopy32(3fml), Fidnm_unload, Fidnm_unload32(3fml)
- Ferror, FERROR32(3fml), Ffind, Ffind32(3fml), Ffindalloc, Ffindalloc32(3fml)
- Ffindocc, Ffindocc32(3fml), Ffindlast, Ffindlast32(3fml), Ffinds, Ffinds32(3fml), Ffloatev, Ffloatev32, Fvfloatev, Fvfloatev32(3fml)
- Ffree, Ffree32(3fml), Fget, Fget32(3fml), Fgets, FGETS32(3Fml)
- Fgetalloc, Fgetalloc32(3fml), Fgetlast, Fgetlast32(3fml), FIDNM_UNLOAD, FIDNM_UNLOAD32(3Fml)
- FIDNM_UNLOAD32(3Fml), Fidnm_used, Fidnm_used32(3fml), Fielded, Fielded32(3fml)
- Findex, Findex32(3fml), Finit, Finit32(3fml), Fjoin, Fjoin32(3fml), Fldid, Fldid32(3fml), Fldno, Fldno32(3fml), Fldtype, Fldtype32(3fml)
- Fielded, Fielded32(3fml), Fielded32(3fml), Fielded32(3fml), Fielded32(3fml)
- Fmove, Fmove32(3fml), Fname, Fname32(3fml), Fneeded, Fneeded32(3fml), Fnext, Fnext32(3fml), Fnmid_unload, Fnmid_unload32(3fml), Fnum, Fnum32(3fml)
- Foccur, Foccur32(3fml), Fjoind, Fjoind32(3fml), Fproj, Fproj32(3fml), Fpres, Fpres32(3fml), Fprint, Fprint32(3fml), Fprojc, Fprojc32(3fml)
- Fprojc, Fprojc32(3fml), Fread, Fread32(3fml), Frealloc, Frealloc32(3fml), Fstrerror, Fstrerror32(3fml), Fstrerror, Fstrerror32(3fml)
- Fstrerror, Fstrerror32(3fml), Ftype, Ftype32(3fml), Funindex, Funindex32(3fml), Funused, Funused32(3fml)
- Fupdate, Fupdate32(3fml), Fused, Fused32(3fml), Fvall, Fvall32(3fml), Fvals, Fvals32(3fml), Fvftsos, FVFTSOS32(3fml)
- Fvftsos, Fvftsos32(3fml), Fneeded, Fneeded32(3fml), Fvnull, Fvnull32(3fml), Fvopt, Fvopt32(3fml), Fvsetinit, Fvsetinit32(3fml)
- Fvsetinit, Fvsetinit32(3fml), Fvstof, Fvstofs32(3fml), Fwrite, Fwrite32(3fml), field_tables(5), viewfile(5)

Programming BEA Tuxedo ATMI Applications Using FML
Section 3fml - FML Functions

CFadd, CFadd32(3fml)

Name  CFadd(), CFadd32() - convert and add field

Synopsis  
#include <stdio.h>  
#include "fml.h"  
int CFadd(FBFR *fbfr, FLID fieldid, char *value, FLDLEN len, int type)  
#include fml32.h>  
int CFadd32(FBFR32 *fbfr, FLDID32 fieldid, char *value, FLDLEN32 len, int type)

Description  CFadd() acts like Fadd() but first converts the value from the user-specified type to  
the type of the fieldid for which the field is added to the fielded buffer. fbfr is a  
pointer to a fielded buffer. fieldid is a field identifier. value is a pointer to the value  
to be added. len is the length of the value to be added; it is required only if type is  
FLD_CARRAY. type is the data type of the field in value.

Before the field is added to the buffer, the type of the data item is converted from the  
type supplied by the user to the type specified in fieldid. If the source type is  
FLD_CARRAY (arbitrary character array), the len argument should be set to the length  
of the array; the length is ignored in all other cases. The value for the field to be  
converted and added must first be put in a variable, value, since C does not permit  
constructs such as 12345L.

This function fails if any of the following field types is used: FLD_PTR, FLD_FML32, or  
FLD_VIEW32. If one of these field types is encountered when CFadd() or CFadd32()  
is being used, Ferror is set to FEBADOP.

CFadd32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to CFadd() or CFadd32()  
while running in any context state, including TPINVALIDCONTEXT.

Return Values  This function returns -1 on error and sets Ferror to indicate the error condition.

Errors  Under the following conditions, CFadd() fails and sets Ferror to:

[FALIGNERR]  
"fielded buffer not aligned"

The buffer does not begin on the proper boundary.
[FNOTFLD]
"buffer not fielded"
The buffer is not a fielded buffer or has not been initialized by Finit().

[FMALLOC]
"malloc failed"
Allocation of space dynamically using malloc() failed when converting from a carray to string.

[FEINVAL]
"invalid argument to function"
One of the arguments to the function invoked was invalid, (for example, a NULL value parameter was specified).

[FNOSPACE]
"no space in fielded buffer"
A field value is to be added or changed in a field buffer, but there is not enough space remaining in the buffer.

[FBADFLD]
"unknown field number or type"
A field identifier is specified which is not valid.

[FTYPERR]
"invalid field type"
A field identifier is specified which is not valid.

[FEBADOP]
"invalid field type"
An invalid field type (such as FLD_PTR, FLD_FML32, and FLD_VIEW32) is specified.

See Also  Introduction to FML Functions, Fadd, Fadd32(3fml)
Section 3fml - FML Functions

CFchg, CFchg32(3fml)

Name  CFchg(), CFchg32() - convert and change field

Synopsis  
#include <stdio.h>
#include "fml.h"
int CFchg(FBFR *fbfr, FLDID fieldid, FLOCC oc, char *value,  
  FLDLEN len, int type)
#include "fml32.h"
int CFchg32(FBFR32 *fbfr, FLDID32 fieldid, FLOCC32 oc,  
  char *value,  
  FLDLEN32 len, int type)

Description  CFchg() acts like Fchg() but first converts the value from the user-specified type  
to the type of the fieldid for which the field is changed in the fielded buffer. fbfr is  
a pointer to a fielded buffer. fieldid is a field identifier. oc is the occurrence number  
of the field. value is a pointer to a new value. len is the length of the value to be  
changed; it is required only if type is FLD_CARRAY. type is the data type of value.

If a field occurrence is specified that does not exist, then NULL values are added for  
the missing occurrences until the desired value can be added (for example, changing  
field occurrence 4 for a field that does not exist in a buffer will cause 3 NULL values  
to be added followed by the specified field value).

This function fails if any of the following field types is used: FLD_PTR, FLD_FML32, or  
FLD_VIEW32. If one of these field types is encountered when CFchg() or CFchg32()  
is being used, Ferror is set to FEBADOP.

CFchg32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to CFchg() or CFchg32()  
while running in any context state, including TPINVALIDCONTEXT.

Return Values  This function returns -1 on error and sets Ferror to indicate the error condition.

Errors  Under the following conditions, CFchg() fails and sets Ferror to:

[FALIGNERR]  
"fielded buffer not aligned"

The buffer does not begin on the proper boundary.
[FNOTFLD]
"buffer not fielded"
The buffer is not a fielded buffer or has not been initialized by Finit().

[F_MALLOC]
"malloc failed"
Allocation of space dynamically using malloc() failed when converting from a carray to string.

[FEINVAL]
"invalid argument to function"
One of the arguments to the function invoked was invalid, (for example, a NULL value parameter was specified).

[FNOSPACE]
"no space in fielded buffer"
A field value is to be added or changed in a field buffer but there is not enough space remaining in the buffer.

[FNOTPRES]
"field not present"
A field occurrence is requested but the specified field and/or occurrence was not found in the fielded buffer.

[FBADFLD]
"unknown field number or type"
A field identifier is specified which is not valid.

[FTYPERR]
"invalid field type"
A field identifier is specified which is not valid.

[FEBADOP]
"invalid field type"
An invalid field type (such as FLD_PTR, FLD_FMML32, and FLD_VIEW32) is specified.

See Also Introduction to FML Functions, CFadd, CFadd32(3fml), Fchg, Fchg32(3fml)
Section 3fml - FML Functions

**CFfind, CFfind32(3fml)**

**Name**  
CFfind(), CFfind32() - find, convert, and return pointer

**Synopsis**  
#include <stdio.h>  
#include "fml.h"  
char * CFfind(FBFR *fbfr, FLDID fieldid, FLDOCC oc, FLDLEN *len, int type)  
#include "fml32.h"  
char * CFfind32(FBFR32 *fbfr, FLDID32 fieldid, FLDOCC32 oc, FLDLEN32 *len, int type)

**Description**  
CFfind() finds a specified field in a buffer, converts it and returns a pointer to the converted value. *fbfr* is a pointer to a fielded buffer. *fieldid* is a field identifier. *oc* is the occurrence number of the field. *len* is used on output and is a pointer to the length of the converted value. *type* is the data type the user wants the field to be converted to.

Like Ffind(), the pointer returned by the function should be considered read-only. The validity of the pointer returned by CFfind() is guaranteed only until the next buffer operation, even if that operation is non-destructive, since the converted value is retained in a single private buffer. This differs from the value returned by Ffins(), which is guaranteed until the next modification of the buffer. Unlike Ffins(), CFfind() aligns the converted value for immediate use by the caller.

This function fails if any of the following field types is used: FLD_PTR, FLD_FMT32, or FLD_VIEW32. If one of these field types is encountered when CFfind() or CFfind32() is being used, Ferror is set to FEBADOP.

CFfind32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to CFfind() or CFfind32() while running in any context state, including TPINVALIDCONTEXT.

**Return Values**  
In the “Synopsis” section above the return value to CFfind() is described as a character pointer data type (char ** in C). Actually, the pointer returned points to an object that has the same type as the stored type of the field.

This function returns NULL on error and sets Ferror to indicate the error condition.

**Errors**  
Under the following conditions, CFfind() fails and sets Ferror to:
The buffer does not begin on the proper boundary.

The buffer is not a fielded buffer or has not been initialized by Finit().

Allocation of space dynamically using malloc() failed when converting from a carray to string.

A field occurrence is requested but the specified field and/or occurrence was not found in the fielded buffer.

A field identifier is specified which is not valid.

A field identifier is specified which is not valid.

An invalid field type (such as FLD_PTR, FLD_FML32, and FLD_VIEW32) is specified.

See Also

Introduction to FML Functions, Ffind, Ffind32(3fml)
## Section 3fml - FML Functions

### CFfindocc, CFfindocc32(3fml)

<table>
<thead>
<tr>
<th>Name</th>
<th>CFfindocc(), CFfindocc32() - find occurrence of converted value</th>
</tr>
</thead>
</table>
| Synopsis | #include <stdio.h>  
#include "fml.h"  
FLDOCC  
CFfindocc(FBFR *fbfr, FLID fieldid, char *value, FLDLEN len, int type)  
#include "fml32.h"  
FLDOCC32  
CFfindocc32(FBFR32 *fbfr, FLID32 fieldid, char *value, FLDLEN32 len, int type) |
| Description | CFfindocc() acts like Findocc() but first converts the value from the user-specified type to the type of fieldid. CFfindocc() looks for an occurrence of the specified field in the buffer that matches a user-supplied value, length and type. CFfindocc() returns the occurrence number of the first field that matches. fbfr is a pointer to a fielded buffer. fieldid is a field identifier. value is a pointer to the value being sought. len is the length of the value to be compared to input value if type is carray. type is the data type of the field in value. This function fails if any of the following field types is used: FLD_PTR, FLD_FML32, or FLD_VIEW32. If one of these field types is encountered when CFfindocc() or CFfindocc32() is being used, Ferror is set to FEBADOP. CFfindocc32() is used with 32-bit FML. A thread in a multithreaded application may issue a call to CFfindocc() or CFfindocc32() while running in any context state, including TPINVALIDCONTEXT. |
| Return Values | If the field value is not found or if other errors are detected, -1 is returned and CFfindocc() sets Ferror to indicate the error condition. |
| Errors | Under the following conditions, CFfindocc() fails and sets Ferror to:  
[FALIGNERR]  
"fielded buffer not aligned"  
The buffer does not begin on the proper boundary.  
[FNOTFLD]  
"buffer not fielded"  
The buffer is not a fielded buffer or has not been initialized by Finit(). |
"malloc failed"
Allocation of space dynamically using malloc() failed when converting from a carray to string.

"invalid argument to function"
One of the arguments to the function invoked was invalid, (for example, a NULL value parameter was specified).

"field not present"
A field occurrence is requested but the specified field and/or occurrence was not found in the fielded buffer.

"unknown field number or type"
A field identifier is specified which is not valid.

"invalid field type"
A field identifier is specified which is not valid.

"invalid field type"
An invalid field type (such as FLD_PTR, FLD_FML32, and FLD_VIEW32) is specified.

See Also
Introduction to FML Functions, Ffindocc, Ffindocc32(3fml)
Section 3fml - FML Functions

CFget, CFget32(3fml)

Name  CFget(), CFget32() - get field and convert

Synopsis  #include <stdio.h>
          #include "fml.h"
          int
          CFget(FBFR *fbfr, FLDID fieldid, FLDOCC oc, char *buf, FLDLEN *len,
          int type)
          #include "fml32.h"
          int
          CFget32(FBFR32 *fbfr, FLDID32 fieldid, FLDOCC32 oc, char *buf,
          FLDLEN32 *len, int type)

Description  CFget() is the conversion analog of Fget(). The main difference is that it copies a
converted value to the user-supplied buffer. fbfr is a pointer to a fielded buffer.
fieldid is a field identifier. oc is the occurrence number of the field. buf is a pointer
to private data area. On input, len is a pointer to the length of the private data area. On
return, len is a pointer to the length of the returned value. If the len parameter is
NULL on input, it is assumed that the buffer is big enough to contain the field value
and the length of the value is not returned. If the buf parameter is NULL, the field
value is not returned. type is the data type the user wants the returned value converted
to.

This function fails if any of the following field types is used: FLD_PTR, FLD_FML32, or
FLD_VIEW32. If one of these field types is encountered when CFget() or CFget32()
is being used, Ferror is set to FEBADOP.

CFget32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to CFget() or CFget32() while
running in any context state, including TPINVALIDCONTEXT.

Return Values  This function returns -1 on error and sets Ferror to indicate the error condition.

Errors  Under the following conditions, CFget() fails and sets Ferror to:

[FALIGNERR]
   "fielded buffer not aligned"
   The buffer does not begin on the proper boundary.
The buffer is not a fielded buffer or has not been initialized by Finit().

Allocation of space dynamically using malloc() failed when converting from a carray to string.

The size of the data area, as specified in len, is not large enough to hold the field value.

A field occurrence is requested but the specified field and/or occurrence was not found in the fielded buffer.

A field identifier is specified which is not valid.

A field identifier is specified which is not valid.

An invalid field type (such as FLD_PTR, FLD_FML32, and FLD_VIEW32) is specified.

Introduction to FML Functions, Fget, Fget32(3fml)
Section 3fml - FML Functions

CFgetalloc, CFgetalloc32(3fml)

Name
CFgetalloc(), CFgetalloc32() - get field, allocate space, convert

Synopsis
#include <stdio.h>
#include "fml.h"

char *
CFgetalloc(FBFR *fbfr, FLDID fieldid, FLDOCC oc, int type, FLDLEN
*extralen)
#include "fml32.h"
char *
CFgetalloc32(FBFR32 *fbfr, FLDID32 fieldid, FLDOCC32 oc, int type,
FLDLEN32 *extralen)

Description
CFgetalloc() gets a specified field from a buffer, allocates space, converts the field
to the type specified by the user and returns a pointer to its location. fbfr is a pointer
to a fielded buffer. fieldid is a field identifier. oc is the occurrence number of the
field. type is the data type the user wants the field to be converted to. On call,
extralen is a pointer to the length of additional space that may be allocated to receive
the value; on return, it is a pointer actual amount of space used. If extralen is NULL,
then no additional space is allocated and the actual length is not returned. The user is
responsible for freeing the returned (converted) value.

This function fails if any of the following field types is used: FLD_PTR, FLD_FML32, or
FLD_VIEW32. If one of these field types is encountered when CFgetalloc() or
CFgetalloc32() is being used, Ferror is set to FEBADOP.

CFgetalloc32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to CFgetalloc() or
CFgetalloc32() while running in any context state, including TPINVALIDCONTEXT.

Return Values
On success, CFgetalloc() returns a pointer to the converted value. On error, the
function returns NULL and sets Ferror to indicate the error condition.

Errors
Under the following conditions, CFgetalloc() fails and sets Ferror to:

[FALIGNERR]
"fielded buffer not aligned"
The buffer does not begin on the proper boundary.
[FNOTFLD]
"buffer not fielded"
The buffer is not a fielded buffer or has not been initialized by Finit().

[FMALLOC]
"malloc failed"
Allocation of space dynamically using malloc() failed.

[FNOTPRES]
"field not present"
A field occurrence is requested but the specified field and/or occurrence was not found in the fielded buffer.

[FBADFLD]
"unknown field number or type"
A field identifier is specified which is not valid.

[FTYPERR]
"invalid field type"
A field identifier is specified which is not valid.

[FEBADOP]
"invalid field type"
An invalid field type (such as FLD_PTR, FLD_FML32, and FLD_VIEW32) is specified.

See Also
Introduction to FML Functions, Fgetalloc, Fgetalloc32(3fml)
F_error, F_error32(3fml)

Name
F_error(), F_error32() - print error message for last error

Synopsis
#include stdio.h>
#include "fml.h"
extern int Ferror;
void
F_error(char *msg)
#include "fml32.h"
extern int Ferror32;
void
F_error32(char *msg)

Description
The function F_error() works like perror() for UNIX system errors; that is, it produces a message on the standard error output (file descriptor 2), describing the last error encountered during a call to a system or library function. The argument string msg is printed first, then a colon and a blank, then the message and a newline. If msg is a NULL pointer or points to a NULL string, the colon is not printed. To be of most use, the argument string should include the name of the program that incurred the error. The error number is taken from the external variable Ferror, which is set when errors occur but not cleared when non-erroneous calls are made. In the MS-DOS and OS/2 environments, Ferror is redefined to FMLerror.

To immediately print an error message, F_error() should be called on an error return from another FML function. When the error message is FEUNIX.Unix_err() is called.

F_error32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to F_error() or F_error32() while running in any context state, including TPINVALIDCONTEXT.

Return Values
F_error() is declared a void and as such does not have return values.

See Also
Introduction to FML Functions

perror(3), Unix_err(3) in a UNIX system reference manual
## F32to16, F16to32(3fml)

<table>
<thead>
<tr>
<th>Name</th>
<th>F32to16(), F16to32() - convert 16-bit FML to/from 32-bit FML buffer</th>
</tr>
</thead>
</table>
| Synopsis | #include <stdio.h>  
#include "fml.h"  
#include "fml32.h"  

int F32to16(FBFR *dest, FBFR32 *src)  
int F16to32(FBFR32 *dest, FBFR *src) |
| Description | F32to16() converts a 32-bit FML buffer to a 16-bit FML buffer. It does this by converting the buffer on a field-by-field basis and then creating the index for the fielded buffer. A field is converted by generating a FLID from a FLID32, and copying the field value (and field length for string and carray fields). dest and src are pointers to the destination and source fielded buffers respectively. The source buffer is not changed.  

These functions can fail for lack of space; they can be reissued after allocating enough additional space to complete the operation.  

F16to32() converts a 16-bit FML buffer to a 32-bit FML buffer. It lives in the.fml32 library or shared object and sets Ferror on error.  

F32to16() lives in the FML library or shared object and sets Ferror on error. Note that both.fml.h and.fml32.h must be included to use these functions; fml1632.h may not be included in the same file.  

F32to16() fails with FBADFLD for the following field types: FLD_PTR, FLD_FML32, and FML_VIEW32. F16to32() has no impact when performed on these field types.  

A thread in a multithreaded application may issue a call to F32to16() or F16to32() while running in any context state, including TPINVALIDCONTEXT. |
| Return Values | This function returns -1 on error and sets Ferror to indicate the error condition. |
| Errors | Under the following conditions, F32to16() fails and sets Ferror to: |
|         | [FALIGNERR]  
|         | "fielded buffer not aligned"  
|         | Either the source buffer or the destination buffer does not begin on the proper boundary. |
Section 3fml - FML Functions

[FNOTFLD]
"buffer not fielded"
Either the source buffer or the destination buffer is not a fielded buffer or has not been initialized by Finit().

[FNOSPACE]
"no space in fielded buffer"
A field value is to be copied to the destination fielded buffer but there is not enough space remaining in the buffer. This error is also returned if a 32-bit FML field is too long to fit into a 16-bit FML field. When this error is returned, the destination buffer will contain no fields.

[FBADFLD]
"invalid field number or type"
This error occurs only for the F32to16() function. The source buffer has a field identifier for which the field type is not one of the eight types supported by 16-bit FML, or the field number is greater than 8191.

See Also Introduction to FML Functions
Fadd, Fadd32(3fml)

Name  
Fadd(), Fadd32() - add new field occurrence

Synopsis  
#include stdio.h>  
#include "fml.h"  
int Fadd(FBFR *fbfr, FLDID fieldid, char *value, FLDLEN len)  
#include "fml32.h"  
int Fadd32(FBFR32 *fbfr, FLDID32 fieldid, char *value, FLDLEN32 len)

Description  
Fadd() adds the specified field value to the given buffer. fbfr is a pointer to a fielded buffer. fieldid is a field identifier. value is a pointer to a new value; the pointer’s type must be the same fieldid type as the value to be added. len is the length of the value to be added; it is required only if type is FLD_CARRAY.

The value to be added is contained in the location pointed to by the value parameter. If one or more occurrences of the field already exist, then the value is added as a new occurrence of the field, and is assigned an occurrence number 1 greater than the current highest occurrence (to add a specific occurrence, Fchg() must be used).

In the “Synopsis” section above the value argument to Fadd() is described as a character pointer data type (char * in C). Technically, this describes only one particular kind of value passable to Fadd(). In fact, the type of the value argument should be a pointer to an object of the same type as the type of the fielded-buffer representation of the field being added. For example, if the field is stored in the buffer as type FLD_LONG, then value should be of type pointer-to-long (long * in C). Similarly, if the field is stored as FLD_SHORT, then value should be of type pointer-to-short (short * in C). The important thing is that Fadd() assumes that the object pointed to by value has the same type as the stored type of the field being added.

For values of type FLD_PTR, Fadd32() stores the pointer value. The buffer pointed to by a FLD_PTR field must be allocated using the tpalloc() call. For values of type FLD_FML32, Fadd32() stores the entire FLD_FML32 field value, except the index. For values of type FLD_VIEW32, Fadd() stores a pointer to a structure of type FVIEWFLD, that contains vflags (a flags field, currently unused and set to 0), vname (a character array containing the viewname), and data (a pointer to the view data stored as a C structure). The application provides the vname and data to Fadd32().
For values of type FLD_CARRAY, the length of the value is given in the \textit{len} argument.
For all types other than FLD_CARRAY, the length of the object referenced by \textit{value} is inferred from its type (for example, a value of type FLD_FLOAT is of length\ sizeof (float)), and the contents of \textit{len} are ignored.

\texttt{Fadd32} is used with 32-bit FML.

A thread in a multithreaded application may issue a call to \texttt{Fadd()} or \texttt{Fadd32()} while running in any context state, including TPINVALIDCONTEXT.

\begin{center}
\textbf{Return Values}
\end{center}

This function returns -1 on error and sets \texttt{Ferror} to indicate the error condition.

\begin{center}
\textbf{Errors}
\end{center}

Under the following conditions, \texttt{Fadd()} fails and sets \texttt{Ferror} to:

\begin{verbatim}
[FALIGNERR]
 "fielded buffer not aligned"
 The buffer does not begin on the proper boundary.

[FNOTFLD]
 "buffer not fielded"
 The buffer is not a fielded buffer or has not been initialized by \texttt{Finit()}.

[FEINVAL]
 "invalid argument to function"
 One of the arguments to the function invoked was invalid. (For example,
 specifying a NULL value parameter to \texttt{Fadd()}).

[FNOSPACE]
 "no space in fielded buffer"
 A field value is to be added in a fielded buffer but there is not enough space
 remaining in the buffer.

[FBADFLD]
 "unknown field number or type"
 A field number is specified which is not valid.

\end{verbatim}

\begin{center}
\textbf{See Also}
\end{center}

Introduction to FML Functions, \texttt{CFadd, CFadd32(3fml), Fadds, Fadds32(3fml), Fchg, Fchg32(3fml)}
Fadds, Fadds32(3fml)

**Name**
Fadds(), Fadds32() - convert value from type FLD_STRING and add to buffer

**Synopsis**
#include <stdio.h>
#include "fml.h"
int
Fadds(FBFR *fbfr, FLDID fieldid, char *value)
#include "fml32.h"
int
Fadds32(FBFR32 *fbfr, FLDID32 fieldid, char *value)

**Description**
Fadds() has been provided to handle the case of conversion from a user type of FLD_STRING to the field type of fieldid and add it to the fielded buffer. fbfr is a pointer to a fielded buffer. fieldid is a field identifier. value is a pointer to the value to be added.

This function calls CFadd() providing a type of FLD_STRING, and a len of 0.

Fadds32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fadds() or Fadds32() while running in any context state, including TPINVALIDCONTEXT.

**Return Values**
This function returns -1 on error and sets Ferror to indicate the error condition.

**Errors**
Under the following conditions, Fadds() fails and sets Ferror to:

[FALIGNERR]
"fielded buffer not aligned"
The buffer does not begin on the proper boundary.

[FNOTFLD]
"buffer not fielded"
The buffer is not a fielded buffer or has not been initialized by Finit().

[FNOSPACE]
"no space in fielded buffer"
A field value is to be added in a fielded buffer but there is not enough space remaining in the buffer.
[FTYPERR]  "invalid field type"
   A field type is specified which is not valid.

[FEINVAL]  "invalid argument to function"
   One of the arguments to the function invoked was invalid, (for example,
   specifying a NULL value parameter to Fadds())

[FMALLOC]  "malloc failed"
   Allocation of space dynamically using malloc() failed during conversion of
   carray to string.

[FBADFLD]  "unknown field number or type"
   A field identifier is specified which is not valid.

See Also  Introduction to FML Functions, CFchg, CFchgs32(3fml), CFfind,
          CFfind32(3fml), CFget, CFget32(3fml), Falloc, Fallocc32(3fml), Fchgs,
          Fchgs32(3fml), Ffinds, Ffinds32(3fml), Fgets, Fgetsa32(3fml), Fgetsa,
**Faloc, Faloc32(3fmi)**

**Name**
Faloc(), Faloc32() - allocate and initialize fielded buffer

**Synopsis**
#include <stdio.h>
#include "fml.h"
FBFR *
Faloc(FLDOCC F, FLDLEN V)
#include "fml32.h"
FBFR32 *
Faloc32(FLDOCC32 F, FLDLEN32 V)

**Description**
Faloc() dynamically allocates space using malloc() for a fielded buffer and calls Finit() to initialize it. The parameters are the number of fields, F, and the number of bytes of value space, V, for all fields that are to be stored in the buffer.

Faloc32() is used for larger buffers with more fields.

A thread in a multithreaded application may issue a call to Faloc() or Faloc32() while running in any context state, including TPINVALIDCONTEXT.

**Return Values**
This function returns NULL on error and sets Ferror to indicate the error condition.

**Errors**
Under the following conditions, Faloc() fails and sets Ferror to:

[FMALLOCS]
"malloc failed"
Allocation of space dynamically using malloc() failed.

[FEINVALS]
"invalid argument to function"
One of the arguments to the function invoked was invalid, (for example, number of fields is less than 0, V is 0 or total size is greater than 65534).

**See Also**
Introduction to FML Functions, Ffree, Ffree32(3fml), Fielded, Fielded32(3fml), Finit, Finit32(3fml), Fneeded, Fneeded32(3fml), Frealloc, Frealloc32(3fml), Fsizeof, Fsizeof32(3fml), Funused, Funused32(3fml)

malloc(3) in a UNIX system reference manual
Section 3fml - FML Functions

Fappend, Fappend32(3fml)

Name
Fappend(), Fappend32() - append new field occurrence

Synopsis
#include <stdio.h>
#include "fml.h"

int Fappend(FBFR *fbfr, FLID fieldid, char *value, FLDLEN len)
#include "fml32.h"

int Fappend32(FBFR32 *fbfr, FLID32 fieldid, char *value, FLDLEN32 len)

Description
Fappend() adds the specified field value to the end of the given buffer. Fappend() is useful in building large buffers in that it does not maintain the internal structures and ordering necessary for general purpose FML access. The side effect of this optimization is that a call to Fappend() may be followed only by additional calls to Fappend(), calls to the FML indexing routines Findex() and Funindex(), or calls to Free(), Fused(), Funused() and Fsizeof(). Calls to other FML routines made before calling Findex() or Funindex() will result in an error with Ferror set to FNOTFLD.

fbfr is a pointer to a fielded buffer. fieldid is a field identifier. value is a pointer to a new value; the pointer’s type must be the same fieldid type as the value to be added. len is the length of the value to be added; it is required only if type is FLD_CARRAY.

The value to be added is contained in the location pointed to by the value parameter. If one or more occurrences of the field already exist, then the value is added as a new occurrence of the field, and is assigned an occurrence number 1 greater than the current highest occurrence (to add a specific occurrence, Fchg() must be used).

In the “Synopsis” section above the value argument to Fappend() is described as a character pointer data type (char * in C). Technically, this describes only one particular kind of value passable to Fappend(). In fact, the type of the value argument should be a pointer to an object of the same type as the type of the fielded-buffer representation of the field being added. For example, if the field is stored in the buffer as type FLD_LONG, then value should be of type pointer-to-long (long * in C). Similarly, if the field is stored as FLD_SHORT, then value should be of type pointer-to-short (short * in C). The important thing is that Fappend() assumes that the object pointed to by value has the same type as the stored type of the field being added.
For values of type FLD_CARRAY, the length of the value is given in the len argument. For all types other than FLD_CARRAY, the length of the object pointed to by value is inferred from its type (for example, a value of type FLD_FLOAT is of length sizeof(float)), and the contents of len are ignored.

Fappend32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fappend() or Fappend32() while running in any context state, including TPINVALIDCONTEXT.

Return Values
This function returns -1 on error and sets Ferror to indicate the error condition.

Errors
Under the following conditions, Fappend() fails and sets Ferror to:

[FALIGNERR]
"fielded buffer not aligned"
The buffer does not begin on the proper boundary.

[FNOTFLD]
"buffer not fielded"
The buffer is not a fielded buffer or has not been initialized by Finit().

[FEINVAL]
"invalid argument to function"
One of the arguments to the function invoked was invalid. (for example, specifying a NULL value parameter to Fappend()).

[FNOSPACE]
"no space in fielded buffer"
A field value is to be added in a fielded buffer but there is not enough space remaining in the buffer.

[FBADFLD]
"unknown field number or type"
A field number is specified which is not valid.

See Also
Introduction to FML Functions, Fadd, Fadd32(3fml), Ffree, Ffree32(3fml), Findex, Findex32(3fml), Fsizeof, Fsizeof32(3fml), Funindex, Funindex32(3fml), Funused, Funused32(3fml), Fused, Fused32(3fml)
Fboolco, Fboolco32, Fvboolco, Fvboolco32(3fml)

Name
Fboolco(), Fboolco32(), Fvboolco(), Fvboolco32() - compile expression, return evaluation tree

Synopsis
#include <stdio.h>
#include "fml.h"
char *
Fboolco(char *expression)
char *
Fboolco32(char *expression)
char *
Fvboolco(char *expression, char *viewname)
#include "fml32.h"
char *
Fvboolco32(char *expression, char *viewname)

Description
Fboolco() compiles a Boolean expression, pointed to by expression, and returns a pointer to the evaluation tree. The expressions recognized are close to the expressions recognized in C. A description of the grammar can be found in the Programming BEA Tuxedo ATMI Applications Using FML.

The evaluation tree produced by Fboolco() is used by the other Boolean functions listed under "See Also:" this avoids having to recompile the expression.

Fboolco32() is used with 32-bit FML.

Fvboolco() and Fvboolco32() provide the same functionality for views. The viewname parameter indicates the view from which the field offsets are taken.

This function fails if any of the following field types is used: FLD_PTR, FLD_FML32, or FLD_VIEW32. If one of these field types is encountered, Ferror is set to FEBADOP.

These functions are not supported on Workstation platforms.

A thread in a multithreaded application may issue a call to any of the functions documented here—Fboolco(), Fboolco32(), Fvboolco(), or Fvboolco32()—while running in any context state, including TPINVALIDCONTEXT.

Return Values
This function returns NULL on error and sets Ferror to indicate the error condition.
Errors

Under the following conditions, Fboolco() fails and sets Ferror to:

[Fmalloc]
"malloc failed"
Allocation of space dynamically using malloc() failed.

[fsyntax]
"bad syntax in Boolean expression"
A syntax error was found in a Boolean expression by Fboolco() other than an unrecognized field name.

[fbadname]
"unknown field name"
A field name is specified which cannot be found in the field tables or viewfiles.

[fEINVAL]
"invalid argument to function"
One of the arguments to the function invoked was invalid, (for example, expression is NULL).

[fbadview]
"cannot find or get view"
viewname was not found in the files specified by VIEWDIR or VIEWFILES.

[fVFopen]
"cannot find or open viewfile"
While trying to find viewname, the program failed to find one of the files specified by VIEWDIR or VIEWFILES.

[eUNIX]
"operating system error"
While trying to find viewname, the program failed to open one of the files specified by VIEWDIR or VIEWFILES for reading.

[fVFSyntax]
"bad viewfile"
While trying to find viewname, one of the files specified by VIEWDIR or VIEWFILES was corrupted or not a viewfile.

[fmalloc]
"malloc failed"
While trying to find viewname, malloc() failed while allocating space to hold the view information.
An invalid field type (such as FLD_PTR, FLD_FMT32, and FLD_VIEW32) is specified.

Example

```c
#include "stdio.h"
#include "fml.h"
extern char *Fboolco();
char *tree;
...
if((tree=Fboolco("FIRSTNAME %% 'J.*n' & SEX = 'M'")) == NULL)
  F_error("pgm_name");
```

This example compiles a Boolean expression that checks if the FIRSTNAME field is in the buffer, begins with 'J' and ends with 'n' (for example, John, Jean, Jurgen, etc.) and the SEX field equal to 'M'.

The first and second characters of the tree array form the least significant byte and the most significant byte, respectively, of an unsigned 16-bit quantity that gives the length, in bytes, of the entire array. This value is useful for copying or otherwise manipulating the array.

See Also Fboolc, Fboolc32, Fvboolc, Fvboolc32(3fml), Fboolp, Fboolpr32, Fvboolp, Fvboolpr32(3fml), Fldid, Fldid32(3fml)
Fboolev, Fboolev32, Fvboolev, Fvboolev32

**Fboolev, Fboolev32, Fvboolev, Fvboolev32(3fml)**

- **Name**
  
  Fboolev(), Fboolev32(), Fvboolev(), Fvboolev32() - evaluate buffer against tree

- **Synopsis**
  
  ```c
  #include stdio.h>
  #include "fml.h"
  int
  Fboolev(FBFR *fbfr, char *tree)
  int
  Fvboolev(char *cstruct, char *tree, char *viewname)
  #include "fml32.h"
  int
  Fboolev32(FBFR32 *fbfr, char *tree)
  int
  Fvboolev32(char *cstruct, char *tree, char *viewname)
  ```

- **Description**
  
  Fboolev() takes a pointer to a fielded buffer, fbfr, and a pointer to the evaluation tree returned from Fboolco(), tree, and returns true (1) if the fielded buffer matches the specified Boolean conditions and false (0) if it does not. This function does not change either the fielded buffer or evaluation tree. The evaluation tree is one previously compiled by Fboolco().

  Fboolev32() is used with 32-bit FML.

  Fvboolev() and Fvboolev32() provide the same functionality for views. The viewname parameter indicates the view from which the field offsets are taken, and should be the same view specified for Fvboolco() or Fvboolco32().

  These functions are not supported on Workstation platforms.

  A thread in a multithreaded application may issue a call to any of the functions documented here—Fboolev(), Fboolev32(), Fvboolev(), or Fvboolev32()—while running in any context state, including TPINVALIDCONTEXT.

- **Return Values**
  
  Fboolev() returns 1 if the expression in the buffer matches the evaluation tree. It returns 0 if the expression fails to match the evaluation tree. This function returns -1 on error and sets Error to indicate the error condition.
Errors Under the following conditions, Fbool ( ) fails and sets Ferror to:

[FALIGNERR]  "fielded buffer not aligned"
The fbfr buffer does not begin on the proper boundary.

[FNOTFLD]    "buffer not fielded"
The fbfr buffer is not a fielded buffer or has not been initialized by Finit ( ).

[FALLOC]     "malloc failed"
Allocation of space dynamically using malloc ( ) failed.

[FEINVAL]    "invalid argument to function"
One of the arguments to the function invoked was invalid, (for example, specifying a NULL tree parameter).

[FSYNTAX]    "bad syntax in Boolean expression"
A syntax error was found in a Boolean expression other than an unrecognized field name.

[FBADVIEW]   "cannot find or get view"
viewname was not found in the files specified by VIEWDIR or VIEWFILES.

[FVFOPEN]    "cannot find or open viewfile"
While trying to find viewname, the program failed to find one of the files specified by VIEWDIR or VIEWFILES.

[EUNIX]      "operating system error"
While trying to find viewname, the program failed to open one of the files specified by VIEWDIR or VIEWFILES for reading.

[FVFSYNTAX]  "bad viewfile"
While trying to find viewname, one of the files specified by VIEWDIR or VIEWFILES was corrupted or not a viewfile.
Fboolv, Fboolv32, Fvboolv, Fvboolv32(3fml)

[FMALLOC]
"malloc failed"
While trying to find viewname, malloc() failed while allocating space to hold the view information.

Example  Using the evaluation tree compiled in the example for Fboolco():

```
#include stdio.h>
#include "fml.h"
#include "fld.tbl.h"
FBFR *fbfr;
...
Fchg(fbfr, FIRSTNAME, 0, "John", 0);
Fchg(fbfr, SEX, 0, "M", 0);
if(Fboolev(fbfr, ttree) > 0)
  fprintf(stderr, "Buffer selected\n");
else
  fprintf(stderr, "Buffer not selected\n");

would print Buffer selected.
```

See Also  Introduction to FML Functions, Fboolco, Fboolco32, Fvboolco, Fvboolco32(3fml), Fboolpr, Fboolpr32, Fvboolpr, Fvboolpr32(3fml)
Section 3fml - FML Functions

Fboolpr, Fboolpr32, Fvboolpr, Fvboolpr32(3fml)

NAME
Fboolpr(), Fboolpr32(), Fvboolpr(), Fvboolpr32() - print Boolean expression as parsed

Synopsis
#include <stdio.h>
#include "fml.h"
void
Fboolpr(char *tree, FILE *iop)
int
Fvboolpr(char *tree, FILE *iop, char *viewname)
#include "fml32.h"
void
Fboolpr32(char *tree, FILE *iop)
int
Fvboolpr32(char *tree, FILE *iop, char *viewname)

Description
Fboolpr() prints a compiled expression to the specified output stream. The evaluation tree, tree, is one previously created with Fboolco(). iop is a pointer of type FILE to the output stream. The output is fully parenthesized, as it was parsed (as indicated by the evaluation tree). The function is useful for debugging.

Fboolpr32() is used with 32-bit FML.

Fvboolpr() and Fvboolpr32() provide the same functionality for views. The viewname parameter indicates the view from which the field offsets are taken, and should be the same view specified for Fvboolco() or Fvboolco32().

These functions are not supported on Workstation platforms.

A thread in a multithreaded application may issue a call to any of the functions documented here—Fboolpr(), Fboolpr32(), Fvboolpr(), or Fvboolpr32()—while running in any context state, including TPINVALIDCONTEXT.

Return Values
Fboolpr() is declared as returning a void, so there are no return values. Fvboolpr() returns -1 if the viewname is not valid.
Errors
Under the following conditions, Fvboolpr() fails and sets Ferror to:

[FBADVIEW]
"cannot find or get view"
viewname was not found in the files specified by VIEWDIR or VIEWFILES.

[FVFOPEN]
"cannot find or open viewfile"
While trying to find viewname, the program failed to find one of the files
specified by VIEWDIR or VIEWFILES.

[EUNIX]
"operating system error"
While trying to find viewname, the program failed to open one of the files
specified by VIEWDIR or VIEWFILES for reading.

[FVFSYNTAX]
"bad viewfile"
While trying to find viewname, one of the files specified by VIEWDIR or
VIEWFILES was corrupted or not a viewfile.

[FMALLOC]
"malloc failed"
While trying to find viewname, malloc() failed while allocating space to
hold the view information.

Portability
This function is not supported using the BEA Tuxedo System Workstation DLL for
Windows.

See Also
Introduction to FML Functions, Fboolco, Fboolco32, Fvboolco,
Fvboolco32(3fml)
Fchg, Fchg32(3fml)

Name
Fchg(), Fchg32() - change field occurrence value

Synopsis
#include <stdio.h>
#include "fml.h"
int
Fchg(FBFR *fbfr, FLID fieldid, FLDOCC oc, char *value, FLDLEN len)
#include "fm132.h"
int
Fchg32(FBFR32 *fbfr, FLID32 fieldid, FLDOCC32 oc, char *value,
        FLDLEN32 len)

Description
Fchg() changes the value of a field in the buffer. fbfr is a pointer to a fielded buffer.
fieldid is a field identifier. oc is the occurrence number of the field. value is a
pointer to a new value, its type must be the same type as the value to be changed (see
below). len is the length of the value to be changed; it is required only if field type is
FLD_CARRAY.

If an occurrence of -1 is specified, then the field value is added as a new occurrence to
the buffer. If the specified field occurrence is found, then the field value is modified to
the value specified. If a field occurrence is specified that does not exist, then NULL
values are added for the missing occurrences until the desired occurrence can be added
(for example, changing field occurrence 4 for a field that does not exist on a buffer will
cause 3 NULL values to be added followed by the specified field value). NULL values
consist of the NULL string (1 byte in length) for string and character values, 0 for long
and short fields, 0.0 for float and double values, and a zero-length string for a character
array. The new or modified value is contained in value and its length is given in len
if it is a character array (ignored in other cases). If value is NULL, then the field
occurrence is deleted. A value to be deleted that is not found, is considered an error.

In the “Synopsis” section above the value argument to Fchg() is described as a
character pointer data type (char * in C). Technically, this describes only one particular
kind of value passable to Fchg(). In fact, the type of the value argument should be a
pointer to an object of the same type as the type of the fielded-buffer representation of
the field being changed. For example, if the field is stored in the buffer as type
FLD_LONG, then value should be of type pointer-to-long (long * in C). Similarly, if
the field is stored as FLD_SHORT, then value should be of type pointer-to-short (short
* in C). The important thing is that Fchg() assumes that the object pointed to by value
has the same type as the stored type of the field being changed.
For values of type FLD_PTR, Fchg32() stores the pointer value. The buffer pointed to by a FLD_PTR field must be allocated using the tpalloc() call. For values of type FLD_FML32, Fchg32() stores the entire FLD_FML32 field value, except the index. For values of type FLD_VIEW32, Fchg() stores a pointer to a structure of type FVIEWFLD, that contains vflags (a flags field, currently unused and set to 0), vname (a character array containing the viewname), and data (a pointer to the view data stored as a C structure). The application provides the vname and data to Fchg32().

Fchg32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fchg() or Fchg32() while running in any context state, including TPINVALIDCONTEXT.

**Return Values**
This function returns -1 on error and sets Ferror to indicate the error condition.

**Errors**
Under the following conditions, Fchg() fails and sets Ferror to:

[FALIGNERR]
"fielded buffer not aligned"
The buffer does not begin on the proper boundary.

[FNOTFLD]
"buffer not fielded"
The buffer is not a fielded buffer or has not been initialized by Finit().

[FNOTPRES]
"field not present"
A field occurrence is requested for deletion but the specified field and/or occurrence was not found in the fielded buffer.

[FNospace]
"no space in fielded buffer"
A field value is to be added or changed in a fielded buffer but there is not enough space remaining in the buffer.

[FBADFLD]
"unknown field number or type"
A field identifier is specified which is not valid.

**See Also**
Introduction to FML Functions, CFchg, CFchg32(3fml), Fadd, Fadd32(3fml), Fcmp, Fcmp32(3fml), Fdel, Fdel32(3fml)
### Section 3fml - FML Functions

#### Fchgs, Fchgs32(3fml)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fchgs(), Fchgs32()</td>
<td>change field occurrence - caller presents string</td>
</tr>
</tbody>
</table>

**Synopsis**

```c
#include <stdio.h>
#include "fml.h"

int Fchgs(FBFR *fbfr, FLDID fieldid, FLDOCC oc, char *value)
#include "fml32.h"

int Fchgs32(FBFR32 *fbfr, FLDID32 fieldid, int oc, char *value)
```

**Description**

Fchgs() is provided to handle the case of conversion from a user type of FLD_STRING. `fbfr` is a pointer to a fielded buffer. `fieldid` is a field identifier. `oc` is the occurrence number of the field. `value` is a pointer to the string to be added. The function calls its non-string-function counterpart, CFchgs(), providing a type of FLD_STRING, and a len of 0 to convert from a string to the field type of `fieldid`. Fchgs32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fchgs() or Fchgs32() while running in any context state, including TPINVALIDCONTEXT.

**Return Values**

This function returns -1 on error and sets Ferror to indicate the error condition.

**Errors**

Under the following conditions, Fchgs() fails and sets Ferror to:

- `[FALIGNERR]` "fielded buffer not aligned"
  The buffer does not begin on the proper boundary.

- `[FNOTFLD]` "buffer not fielded"
  The buffer is not a fielded buffer or has not been initialized by Finit().

- `[FNOSPACE]` "no space in fielded buffer"
  A field value is to be added or changed in a fielded buffer but there is not enough space remaining in the buffer.

- `[FBADFLD]` "unknown field number or type"
  A field identifier is specified which is not valid.
Fchg, Fchgs32(3fml)

[FTYPERR]
"invalid field type"
A field identifier is specified which is not valid.

See Also Introduction to FML Functions, CFchg, CFchgs32(3fml), Fchg, Fchgs32(3fml)
Section 3fml - FML Functions

Fchksum, Fchksum32(3fml)

Name
Fchksum(), Fchksum32() - compute checksum for fielded buffer

Synopsis
#include <stdio.h>
#include "fml.h"
long
Fchksum(FBFR *fbfr)
#include "fml32.h"
long
Fchksum32(FBFR32 *fbfr)

Description
For extra-reliable I/O, a checksum may be calculated using Fchksum() and stored in a fielded buffer being written out. fbfr is a pointer to a fielded buffer. The stored checksum may be inspected by the receiving process to verify that the entire buffer was received.

For values of type FLD_PTR, the name of the pointer field (rather than the pointer or the data referenced by the pointer) is included in the checksum calculation.

Fchksum32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fchksum() or Fchksum32() while running in any context state, including TPINVALIDCONTEXT.

Return Values
On success, Fchksum() returns the checksum. This function returns -1 on error and sets Ferror to indicate the error condition.

Errors
Under the following conditions, Fchksum() fails and sets Ferror to:

[FALIGNERR]
"fielded buffer not aligned"
The buffer does not begin on the proper boundary.

[FNOTFLD]
"buffer not fielded"
The buffer is not a fielded buffer or has not been initialized by Finit().

See Also
Introduction to FML Functions, Fread, Fread32(3fml), Fwrite, Fwrite32(3fml)
**Fcmp, Fcmp32(3fml)**

**Name**
Fcmp(), Fcmp32() - compare two fielded buffers

**Synopsis**
```
#include <stdio.h>
#include "fml.h"
int
Fcmp(FBFR *fbfr1, FBFR *fbfr2)
#include "fml32.h"
int
Fcmp32(FBFR32 *fbfr1, FBFR32 *fbfr2)
```

**Description**
Fcmp() compares the field identifiers and then the field values of two FML buffers. 

*fbfr1* and *fbfr2* are pointers to the fielded buffers to be compared.

For values of type FLD_PTR, two pointer fields are considered equal if the pointer values (addresses) are equal. For values of type FLD_FML32, two fields are considered equal if all field occurrences and values are equal. For values of type FLD_VIEW32, two fields are considered equal if the viewnames are the same, and if all structure member occurrences and values are equal.

Fcmp32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fcmp() or Fcmp32() while running in any context state, including TPINVALIDCONTEXT.

**Return Values**
The function returns a 0 if the two buffers are identical. It returns a -1 on any of the following conditions:
- The fieldid of a *fbfr1* field is less than the fieldid of the corresponding field of *fbfr2*.
- The value of a field in *fbfr1* is less than the value of the corresponding field of *fbfr2*.
- *fbfr1* has fewer fields or field occurrences than *fbfr2*.

Fcmp() returns a 1 if any of the reverse set of conditions is true, for example, the fieldid of a *fbfr1* field is greater than the fieldid of the corresponding field of *fbfr2*. The actual sizes of the buffers (that is, the sizes passed to Falloc()) are not considered; only the data in the buffers. This function returns -2 on error and sets Ferror to indicate the error condition.
Section 3fml - FML Functions

Errors  Under the following conditions, Fcmp() fails and sets Ferror to:

[FALIGNERR]
  "fielded buffer not aligned"
  The buffer does not begin on the proper boundary.

[FNOTFLD]
  "buffer not fielded"
  The buffer is not a fielded buffer or has not been initialized by Finit().

See Also  Introduction to FML Functions, Fadd, Fadd32(3fml), Fchg, Fchg32(3fml)
Fconcat, Fconcat32(3fml)

**Name**  
Fconcat(), Fconcat32() - concatenate source to destination buffer

**Synopsis**  
```c
#include <stdio.h>
#include "fml.h"
int Fconcat(FBFR *dest, FBFR *src)
#include "fml32.h"
int Fconcat32(FBFR32 *dest, FBFR32 *src)
```

**Description**  
Fconcat() adds fields from the source buffer to the fields that already exist in the destination buffer. dest and src are pointers to the destination and source fielded buffers, respectively. Occurrences in the destination buffer, if any, are maintained and new occurrences from the source buffer are added with greater occurrence numbers for the field.

Fconcat32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fconcat() or Fconcat32() while running in any context state, including TPINVALIDCONTEXT.

**Return Values**  
This function returns -1 on error and sets Ferror to indicate the error condition.

**Errors**  
Under the following conditions, Fconcat() fails and sets Ferror to:

- [FALIGNERR]  
  "fielded buffer not aligned"
  Either the source buffer or the destination buffer does not begin on the proper boundary.

- [FNOTFLD]  
  "buffer not fielded"
  Either the source or the destination buffer is not a fielded buffer or has not been initialized by Finit().

- [FNOSPACE]  
  "no space in fielded buffer"
  A field value is to be added in a fielded buffer but there is not enough space remaining in the buffer.

**See Also**  
Introduction to FML Functions, Fjoin, Fjoin32(3fml), Fupdate, Fupdate32(3fml)
**Section 3fml - FML Functions**

### Fcpy, Fcpy32(3fml)

**Name**
Fcpy(), Fcpy32() - copy source to destination buffer

**Synopsis**
```
#include <stdio.h>
#include "fml.h"
int Fcpy(FBFR *dest, FBFR *src)
#include "fml32.h"
int Fcpy32(FBFR32 *dest, FBFR32 *src)
```

**Description**
Fcpy() is used to copy the contents of one fielded buffer to another fielded buffer. dest and src are pointers to the destination and source fielded buffers respectively. Fcpy() expects the destination to be a fielded buffer, and thus can check that it is large enough to accommodate the data from the source buffer.

For values of type FLD_PTR, Fcpy32() copies the buffer pointer. The application programmer must manage the reallocation and freeing of buffers when the associated pointer is copied.

Fcpy32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fcpy() or Fcpy32() while running in any context state, including TPINVALIDCONTEXT.

**Return Values**
This function returns -1 on error and sets Ferror to indicate the error condition.

**Errors**
Under the following conditions, Fcpy() fails and sets Ferror to:

[FALIGNERR]
"fielded buffer not aligned"
Either the source buffer or the destination buffer does not begin on the proper boundary.

[FNOTFLD]
"buffer not fielded"
Either the source or the destination buffer is not a fielded buffer or has not been initialized by Finit().

[FNSPACE]
"no space in fielded buffer"
The destination buffer is not large enough to hold the source buffer.

**See Also**
Introduction to FML Functions, Fmove, Fmove32 (3fml)

50 ATMI FML Function Reference
**Fdel, Fdel32(3fml)**

**Name**  
Fdel(), Fdel32() - delete field occurrence from buffer

**Synopsis**  
#include stdio.h>
#include "fml.h"
int Fdel(FBFR *fbfr, FLDID fieldid, FLDOCC oc)
#include "fml32.h"
int Fdel32(FBFR32 *fbfr, FLDID32 fieldid, FLDOCC32 oc)

**Description**  
Fdel() deletes the specified field occurrence from the buffer. fbfr is a pointer to a fielded buffer. fieldid is a field identifier. oc is the occurrence number of the field.

Note that when multiple occurrences of a field exist in the fielded buffer and a field occurrence is deleted that is not the last occurrence, also higher occurrences in the buffer are shifted down by one. To maintain the same occurrence number for all occurrences, use Fchg() to set the field occurrence value to a NULL value.

For values of type FLD_PTR, Fdel32() deletes the FLD_PTR field occurrence without changing the referenced buffer or freeing the pointer. The data buffer is treated as an opaque pointer.

Fdel32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fdel() or Fdel32() while running in any context state, including TPINVALIDCONTEXT.

**Return Values**  
This function returns -1 on error and sets Ferror to indicate the error condition.

**Errors**  
Under the following conditions, Fdel() fails and sets Ferror to:

[FALIGNERR]  
"fielded buffer not aligned"
  The buffer does not begin on the proper boundary.

[FNOTFIELD]  
"buffer not fielded"
  The buffer is not a fielded buffer or has not been initialized by Finit().
Section 3fml - FML Functions

[FNOTPRES]
"field not present"
A field occurrence is requested but the specified field and/or occurrence was not found in the fielded buffer.

[FBADFLD]
"unknown field number or type"
A field identifier is specified which is not valid.

See Also: Introduction to FML Functions, Fadd, Fadd32(3fml), Fchg, Fchg32(3fml), Fdelall, Fdelall32(3fml), Fdelete, Fdelete32(3fml)
**Fdelall, Fdelall32(3fml)**

**Name**
Fdelall(), Fdelall32() - delete all field occurrences from buffer

**Synopsis**
```
#include <stdio.h>
#include "fml.h"
int Fdelall(FBFR *fbfr, FLDID fieldid)
#include "fml32.h"
int Fdelall32(FBFR32 *fbfr, FLDID32 fieldid)
```

**Description**
Fdelall() deletes all occurrences of the specified field in the buffer. *fbfr* is a pointer to a fielded buffer. *fieldid* is a field identifier. If no occurrences of the field are found, it is considered an error.

For values of type FLD_PTR, Fdelall32() deletes the FLD_PTR field occurrence without changing the referenced buffer or freeing the pointer. The data buffer is treated as an opaque pointer.

Fdelall32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fdelall() or Fdelall32() while running in any context state, including TPINVALIDCONTEXT.

**Return Values**
This function returns -1 on error and sets Ferror to indicate the error condition.

**Errors**
Under the following conditions, Fdelall() fails and sets Ferror to:

- **[FALIGNERR]**
  "fielded buffer not aligned"
  The buffer does not begin on the proper boundary.

- **[FNOTFLD]**
  "buffer not fielded"
  The buffer is not a fielded buffer or has not been initialized by Finit().

- **[FNOTPRES]**
  "field not present"
  A field is requested but the specified field was not found in the fielded buffer.
Section 3fml - FML Functions

[FBADFLD]
"unknown field number or type"
A field identifier is specified which is not valid.

See Also
Introduction to FML Functions, Fdel, Fdel32(3fml), Fdelete, Fdelete32(3fml)
Fdelete, Fdelete32(3fml)

Name
Fdelete(), Fdelete32() - delete list of fields from buffer

Synopsis
#include <stdio.h>
#include "fml.h"
int
Fdelete(FBFR *fbfr, FLDID *fieldid)
#include "fml32.h"
int
Fdelete32(FBFR32 *fbfr, FLDID32 *fieldid)

Description
Fdelete() deletes all occurrences of all fields listed in the array of field identifiers, fieldid[]. The last entry in the array must be BADFLDID. fbfr is a pointer to a fielded buffer. fieldidis a pointer to an array of field identifiers. This is a more efficient way of deleting several fields from a buffer instead of using several Fdelall() calls. The update is done in-place. The array of field identifiers may be rearranged by Fdelete() (they are sorted, if not already, in numeric order).

For values of type FLD_PTR, Fdelete32() deletes the FLD_PTR field occurrence without changing the referenced buffer or freeing the pointer. The data buffer is treated as an opaque pointer.

Fdelete() returns success even if no fields are deleted from the fielded buffer.

Fdelete32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fdelete() or Fdelete32() while running in any context state, including TPINVALIDCONTEXT.

Return Values
This function returns -1 on error and sets Ferror to indicate the error condition.

Errors
Under the following conditions, Fdelete() fails and sets Ferror to:

[FALIGNERR]
"fielded buffer not aligned"
The buffer does not begin on the proper boundary.

[FNOTFLD]
"buffer not fielded"
The buffer is not a fielded buffer or has not been initialized by Finit().
[FBADFLD]
"unknown field number or type"
A field identifier is specified which is not valid.

See Also
Introduction to FML Functions, Fdel, Fdel32(3fml), Fdelall, Fdelall32(3fml)
# Fextread, Fextread32(3fml)

## Name
Fextread(), Fextread32() - build fielded buffer from printed format

## Synopsis
```c
#include <stdio.h>
#include "fml.h"
int Fextread(FBFR *fbfr, FILE *iop)
#include "fml32.h"
int Fextread32(FBFR32 *fbfr, FILE *iop)
```

## Description
Fextread() may be used to construct a fielded buffer from its printed format (that is, from the output of Fprint()). The parameters are a pointer to a fielded buffer, `fbfr`, and a pointer to a file stream, `iop`. The input file format is basically the same as the output format of `Fprint()`, that is:

```
[flag] fldname or fidid tab> fldval (or fldname, if flag is `\``=``) 
```

The optional flags and their meanings are as follows:

```
+
  Occurrence 0 of the field in the fielded buffer should be changed to the value provided.

-
  Occurrence 0 of the field named should be deleted from the fielded buffer.
  The tab character is required; any field value is ignored.

=
  In this case, the last field on the input line is the name of a field in the fielded buffer. The value of occurrence 0 of that field should be assigned to occurrence 0 of the first field named on the input line.

#
  The line is treated as a comment and is ignored.
```

If no `flag` is specified, a new occurrence of the field named by `fldname` with value `fldval` is added to the fielded buffer. A trailing newline (`\n`) must be provided after each completed input buffer.
For values of type FLD_FML32 and FLD_VIEW32, Fextread32() generates nested FML32 buffers and VIEW32 fields, respectively. This function ignores the FLD_PTR field type. No error is returned if a value of type FLD_PTR is supplied to the function.

Fextread32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fextread() or Fextread32() while running in any context state, including TPINVALIDCONTEXT.

**Return Values**
This function returns -1 on error and sets Ferror to indicate the error condition.

**Errors**
Under the following conditions, Fextread() fails and sets Ferror to:

[FALIGNERR]
"fielded buffer not aligned"
The buffer does not begin on the proper boundary.

[FNOTFLD]
"buffer not fielded"
The buffer is not a fielded buffer or has not been initialized by Finit().

[FNOSPACE]
"no space in fielded buffer"
A field value is to be added or changed in a field buffer but there is not enough space remaining in the buffer.

[FBADFLD]
"unknown field number or type"
A field number is specified which is not valid.

[FEUNIX]
"UNIX system call error"
A UNIX system call error occurred. The external integer errno should have been set to indicate the error by the system call, and the external integer Unixerr (values defined in Unix.h) is set to the system call that returned the error.

[FBADNAME]
"unknown field name"
A field name is specified which cannot be found in the field tables.

[FSYNTAX]
"bad syntax in format"
A syntax error was found in the external buffer format. Possible errors are: an
unexpected end-of-file indicator, input lines not in the form fieldid or name
tab> value two control characters, field values greater than 1000 characters,
or an invalid hex escape sequence.

[FNOTPRES]
"field not present"
A field to be deleted is not found in the fielded buffer.

[FMALLOC]
"malloc failed"
Allocation of space dynamically using malloc() failed.

[FEINVAL]
"invalid parameter"
The value of iop is NULL.

Portability This function is not supported using the BEA Tuxedo System Workstation DLL for Windows.

See Also Introduction to FML Functions, Fprint, Fprint32(3fml)
**Ffind, Ffind32(3fml)**

**Name**
Ffind, Ffind32 - find field occurrence in buffer

**Synopsis**
```c
#include <stdio.h>
#include "fml.h"

char *
Ffind(FBFR *fbfr, FLDID fieldid, FLDOCC oc, FLDLEN *len)
#include "fml32.h"

char *
Ffind32(FBFR32 *fbfr, FLDID32 fieldid, FLDOCC32 oc, FLDLEN32 *len)
```

**Description**
Ffind() finds the value of the specified field occurrence in the buffer. fbfr is a pointer to a fielded buffer. fieldid is a field identifier. oc is the occurrence number of the field. If the field is found, its length is set into *len, and its location is returned as the value of the function. If the value of len is NULL, then the field length is not returned. Ffind() is useful for gaining read-only access to a field. In no case should the value returned by Ffind() be used to modify the buffer.

In general, because proper alignment within a buffer is not guaranteed, the locations in which the values of types FLD_LONG, FLD_FLOAT, FLD_DOUBLE, FLD_PTR, FLD_FML32, and FLD_VIEW32 are stored prevents these values from being used directly as their stored type. Such values must be copied first to a suitably aligned memory location. Accessing such fields through the conversion function CfFfind() does guarantee the proper alignment of the found converted value. Buffer modification should be done only by the Fadd() or Fchg() function. The values returned by Ffind() and Ffind32() are valid only so long as the buffer remains unmodified.

Ffind32() does not check for occurrences of the specified field in embedded buffers as provided by the FLD_FML32 and FLD_VIEW32 field types.

Ffind32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Ffind() or Ffind32() while running in any context state, including TP_INVALIDCONTEXT.

**Return Values**
In the “Synopsis” section above the return value to Ffind() is described as a character pointer data type (char * in C). Actually, the pointer returned points to an object that has the same type as the stored type of the field.

This function returns a pointer to NULL on error and sets Ferror to indicate the error condition.
Errors

Under the following conditions, Ffind() fails and sets Ferror to:

[FALIGNERR]
"fielded buffer not aligned"
The buffer does not begin on the proper boundary.

[FNOTFLD]
"buffer not fielded"
The buffer is not a fielded buffer or has not been initialized by Finit().

[FNOTPRES]
"field not present"
A field occurrence is requested but the specified field and/or occurrence was not found in the fielded buffer.

[FBADFLD]
"unknown field number or type"
A field identifier is specified which is not valid.

See Also

Introduction to FML Functions, Ffindlast, Ffindlast32(3fml), Ffindocc, Ffindocc32(3fml), Ffinds, Ffinds32(3fml)
Section 3fml - FML Functions

Ffindlast, Ffindlast32(3fml)

Name

Ffindlast(), Ffindlast32() - find last occurrence of field in buffer

Synopsis

#include <stdio.h>
#include "fml.h"

char *
Ffindlast(FBFR *fbfr, FLDID fieldid, FLDOCC *oc, FLDLEN *len)

#include "fml32.h"

char *
Ffindlast32(FBFR32 *fbfr, FLDID32 fieldid, FLDOCC32 *oc, FLDLEN32
*len)

Description

Ffindlast() finds the last occurrence of a field in a buffer. fbfr is a pointer to a
fielded buffer. fieldid is a field identifier. oc is a pointer to an integer that is used to
receive the occurrence number of the field. len is the length of the value. If there are
no occurrences of the field in the buffer, NULL is returned. Generally, Ffindlast() acts like Ffind(). The major difference is that with Ffindlast the user does not supply a field occurrence. Instead, both the value and occurrence number of the last occurrence of the field are returned. In order to return the occurrence number of the last field, the occurrence argument, oc, to Ffindlast() is a pointer-to-integer, and not an integer, as it is to Ffind(). If oc is specified to be NULL, the occurrence number of the last occurrence is not returned. If the value of len is NULL, then the field length is not returned.

In general, because proper alignment within a buffer is not guaranteed, the locations in
which the values of types FLD_LONG, FLD_FLOAT, FLD_DOUBLE, FLD_PTR,
FLD_FML32, and FLD_VIEW32 are stored prevents these values from being used
directly as their stored type. Such values must be copied first to a suitably aligned
memory location. Accessing such fields through the conversion function Ffind() does guarantee the proper alignment of the found converted value. Buffer modification
should be done only by the Fadd() or Fchg() function. The values returned by
Ffind() and Ffindlast() are valid only so long as the buffer remains unmodified.

Ffindlast32() does not check for occurrences of the specified field in embedded
buffers as provided by the FLD_FML32 and FLD_VIEW32 field types.

Ffindlast32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Ffindlast() or
Ffindlast32() while running in any context state, including TP_INVALIDCONTEXT.

62 ATMI FML Function Reference
Return Values

In the “Synopsis” section above the return value to `Ffindlast()` is described as a character pointer data type (`char *` in C). Actually, the pointer returned points to an object that has the same type as the stored type of the field.

This function returns NULL on error and sets `Ferror` to indicate the error condition.

Errors

Under the following conditions, `Ffindlast()` fails and sets `Ferror` to:

[FALIGNERR]  
"fielded buffer not aligned"
The buffer does not begin on the proper boundary.

[FNOTFLD]  
"buffer not fielded"
The buffer is not a fielded buffer or has not been initialized by `Finit()`.

[FNOTPRES]  
"field not present"
A field is requested but the specified field was not found in the fielded buffer.

[FBADFLD]  
"unknown field number or type"
A field identifier is specified which is not valid.

See Also

Introduction to FML Functions, 
`CPfind`, `CPfind32(3fml)`, `Fadd`, `Fadd32(3fml)`, 
`Fchg`, `Fchg32(3fml)`, `Ffind`, `Ffind32(3fml)`, `Ffindocc`, `Ffindocc32(3fml)`, 
`Ffinds`, `Ffinds32(3fml)`
Section 3fml - FML Functions

Ffindocc, Ffindocc32(3fml)

Name  Ffindocc(), Ffindocc32() - find occurrence of field value

Synopsis  

#include <stdio.h>
#include "fml.h"

Ffindocc(FBFR *fbfr, FLDID fieldid, char *value, FLDLEN len)
#define include "fm132.h"

Ffindocc32(FBFR32 *fbfr, FLDID32 fieldid, char *value, FLDLEN32 len)

Description  Ffindocc() looks at occurrences of the specified field in the buffer and returns the occurrence number of the first field occurrence that matches the user-specified field value. fbfr is a pointer to a fielded buffer. fieldid is a field identifier. The value to be found is contained in the location pointed to by the value parameter. len is the length of the value if its type is FLD_CARRAY. If fieldid is field type FLD_STRING and if len is not 0, pattern matching is done on the string. The pattern match supported is the same as the patterns described in regcmp(3) (in UNIX reference manuals). In addition, the alternation of regular expressions is supported (for example, "A|B" matches with "A" or "B"). The pattern must match the entire field value (that is, the pattern "value" is implicitly treated as "value"). The version of Ffindocc() provided for use in the MS-DOS and OS/2 environments does not support the regcmp() pattern matching for FLD_STRING fields; it uses strcmp() (in UNIX reference manuals).

In the “Synopsis” section above the value argument to Ffindocc() is described as a character pointer data type (char * in C). Technically, this describes only one particular kind of value passable to Ffindocc(). In fact, the type of the value argument should be a pointer to an object of the same type as the type of the fielded-buffer representation of the field being found. For example, if the field is stored in the buffer as type FLD_LONG, then value should be of type pointer-to-long (long * in C). Similarly, if the field is stored as FLD_SHORT, then value should be of type pointer-to-short (short * in C). The important thing is that Ffindocc() assumes that the object pointed to by value has the same type as the stored type of the field being found.
For values of type FLD_PTR, Ffindocc32() finds the occurrence of a field that matches a specified pointer value. For values of type FLD_FML32, two fields are considered equal if all field occurrences and values are equal. For values of type FLD_VIEW32, two fields are considered equal if the viewnames are the same, and if all structure member occurrences and values are equal.

Ffindocc32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Ffindocc() or Ffindocc32() while running in any context state, including TPINVALIDCONTEXT.

### Return Values

This function returns -1 on error and sets Ferror to indicate the error condition.

### Errors

Under the following conditions, Ffindocc() fails and sets Ferror to:

- **FALIGNERR**
  - "fielded buffer not aligned"
  - The buffer does not begin on the proper boundary.

- **FNOTFLD**
  - "buffer not fielded"
  - The buffer is not a fielded buffer or has not been initialized by Finit().

- **FNOTPRES**
  - "field not present"
  - A field value is requested but the specified field and/or value was not found in the fielded buffer.

- **FEINVAL**
  - "invalid argument to function"
  - One of the arguments to the function invoked was invalid, (for example, passing a NULL value parameter to Ffindocc() or specifying an invalid string pattern).

- **FBADFLD**
  - "unknown field number or type"
  - A field identifier is specified which is not valid.

### See Also

Introduction to FML Functions, Ffind, Ffind32(3fml), Ffindlast, Ffindlast32(3fml), Ffinds, Ffinds32(3fml)

regcmp(3) in a UNIX system reference manual
Section 3fml - FML Functions

Ffinds, Ffinds32(3fml)

Name
Ffinds(), Ffinds32() - return ptr to string representation

Synopsis
#include <stdio.h>
#include "fml.h"
char *
Ffinds(FBFR *fbfr, FLDID fieldid, FLDOCC oc)
#include "fml32.h"
char *
Ffinds32(FBFR32 *fbfr, FLDID32 fieldid, FLDOCC32 oc)

Description
Ffinds() is provided to handle the case of conversion to a user type of FLD_STRING. fbfr is a pointer to a fielded buffer, fieldid is a field identifier, oc is the occurrence number of the field. The specified field occurrence is found and converted from its type in the buffer to a NULL-terminated string. Basically, this macro calls its conversion function counterpart, CFfind(), providing a utype of FLD_STRING, and aulen of 0. The duration of the validity of the pointer returned by Ffinds() is the same as that described for CFfind().

Ffinds32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Ffinds() or Ffinds32() while running in any context state, including TPINVALIDCONTEXT.

Return Values
This function returns NULL on error and sets Ferror to indicate the error condition.

Errors
Under the following conditions, Ffinds() fails and sets Ferror to:

[FALIGNERR]
"fielded buffer not aligned"
The buffer does not begin on the proper boundary.

[FNOTFLD]
"buffer not fielded"
The buffer is not a fielded buffer or has not been initialized by Finit().

[FNOTPRES]
"field not present"
A field occurrence is requested but the specified field and/or occurrence was not found in the fielded buffer.
[FBADFLD]
"unknown field number or type"
A field identifier is specified which is not valid.

[FTYPERR]
"invalid field type"
A field type is specified which is not valid.

[FMALLOC]
"malloc failed"
Allocation of space dynamically using malloc() failed while converting carray to string.

See Also  Introduction to FML Functions, CFfind, CFfind32(3fml), Ffind, Ffind32(3fml)
Ffloatev, Ffloatev32, Fvfloatev, Fvfloatev32(3fml)

Name  
Ffloatev(), Ffloatev32(), Fvfloatev(), Fvfloatev32() - return value of expression as a double

Synopsis  
#include <stdio.h>
#include "fml.h"
double Ffloatev(FBFR *fbfr, char *tree)  
double Fvfloatev(char *cstruct, char *tree, char *viewname)  
#include "fml32.h"
double Ffloatev32(FBFR32 *fbfr, char *tree)  
double Fvfloatev32(char *cstruct, char *tree, char *viewname)

Description  
Ffloatev() takes a pointer to a fielded buffer, fbfr, and a pointer to the evaluation tree returned from Fboolco(), tree, and returns the value of the (arithmetic) expression, represented by the tree, as a double. This function does not change either the fielded buffer or the evaluation tree.

Ffloatev32() is used with 32-bit FML.

Fvfloatev() and Fvfloatev32() provide the same functionality for views. The viewname parameter indicates the view from which the field offsets are taken, and should be the same view specified for Fboolco() or Fboolco32().

These functions are not supported on Workstation platforms.

A thread in a multithreaded application may issue a call to any of the functions documented here—Ffloatev(), Ffloatev32(), Fvfloatev(), or Fvfloatev32()—while running in any context state, including TPINVALIDCONTEXT.

Return Values  
On success Ffloatev() returns the value of an expression as a double.

This function returns -1 on error and sets Ferror to indicate the error condition.
Errors

Under the following conditions, Ffloatev() fails and sets Ferror to:

[FALIGNERR]
"fielded buffer not aligned"
The buffer does not begin on the proper boundary.

[FNOTFLD]
"buffer not fielded"
The buffer is not a fielded buffer or has not been initialized by Finit().

[FMALLOC]
"malloc failed"
Allocation of space dynamically using malloc() failed.

[FSYNTAX]
"bad syntax in Boolean expression"
A syntax error was found in a Boolean expression tree.

[FBADVIEW]
"cannot find or get view"
viewname was not found in the files specified by VIEWDIR or VIEWFILES.

[FVFOPEN]
"cannot find or open viewfile"
While trying to find viewname, the program failed to find one of the files specified by VIEWDIR or VIEWFILES.

[EUNIX]
"operating system error"
While trying to find viewname, the program failed to open one of the files specified by VIEWDIR or VIEWFILES for reading.

[FVFSYNTAX]
"bad viewfile"
While trying to find viewname, one of the files specified by VIEWDIR or VIEWFILES was corrupted or not a viewfile.

[FMALLOC]
"malloc failed"
While trying to find viewname, malloc() failed while allocating space to hold the view information.

See Also
Introduction to FML Functions, Fboolco, Fboolco32, Fvboolco, Fvboolco32(3fml), Fboolev, Fboolev32, Fvboolev, Fvboolev32(3fml)
Section 3fml - FML Functions

Ffprint, Ffprint32(3fml)

Name
Ffprint(), Ffprint32() - print fielded buffer to specified stream

Synopsis
#include <stdio.h>
#include "fml.h"
int
Ffprint(FBFR *fbfr, FILE *iop)
#include "fml32.h"
int
Ffprint32(FBFR32 *fbfr, FILE *iop)

Description
Ffprint() is similar to Fprint(). except the text is printed to a specified output stream. fbfr is a pointer to a fielded buffer. iop is a pointer of type FILE that points to the output stream.

For each field in the buffer, the output prints the field name and field value separated by a tab. Fname() is used to determine the field name; if the field name cannot be determined, then the field identifier is printed. Non-printable characters in string and character array field values are represented by a backslash followed by their two-character hexadecimal value. A newline is printed following the output of the printed buffer.

For values of type FLD_PTR, Ffprint32() prints the field name or field identifier and the pointer value in hexadecimal. Although this function prints pointer information, the Fextread32() function ignores the FLD_PTR field type.

For values of type FLD_FML32, Ffprint32() recursively prints the FML32 buffer, with leading tabs added for each level of nesting. For values of type FLD_VIEW32, Ffprint32() prints the VIEW32 field name and structure member name/value pairs.

Ffprint32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Ffprint() or Ffprint32() while running in any context state, including TP_INVALIDCONTEXT.

Return Values
This function returns -1 on error and sets Ferror to indicate the error condition.

Errors
Under the following conditions, Ffprint() fails and sets Ferror to:

[FALIGNERR]
"fielded buffer not aligned"
The buffer does not begin on the proper boundary.
**Fprint, Fprint32(3fml)**

[FNOTFLD]
"buffer not fielded"
The buffer is not a fielded buffer or has not been initialized by Finit().

[FMALLOC]
"malloc failed"
Allocation of space dynamically using malloc() failed.

**Portability**
This function is not supported using the BEA Tuxedo System Workstation DLL for Windows.

**See Also**
Introduction to FML Functions, Fprint, Fprint32(3fml)
Section 3fml - FML Functions

Ffree, Ffree32(3fml)

Name
Ffree(), Ffree32() - free space allocated for fielded buffer

Synopsis
#include <stdio.h>
#include "fml.h"
int
Ffree(FBFR *fbfr)
#include "fml32.h"
int
Ffree32(FBFR32 *fbfr)

Description
Ffree() is used to recover space allocated to its argument fielded buffer. fbfr is a
pointer to a fielded buffer. The fielded buffer is invalidated, that is, it is made
non-fielded, and then freed. Ffree32() does not free the memory area referenced by
a pointer in a FLD_PTR field.

Ffree() is recommended as opposed to free() (in UNIX system reference manuals),
because Ffree() invalidates a fielded buffer whereas free() does not. It is important
to invalidate fielded buffers because malloc() (in UNIX system reference manuals)
reuses memory that has been freed without clearing it. Thus, if free() were used, it
would be possible for malloc() to return a piece of memory that looks like a valid
fielded buffer but is not.

Ffree32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Ffree() or Ffree32()
while running in any context state, including TPINVALIDCONTEXT.

Return Values
This function returns -1 on error and sets Ferror to indicate the error condition.

Errors
Under the following conditions, Ffree() fails and sets Ferror to:

[FALIGNERR]
"fielded buffer not aligned"
The buffer does not begin on the proper boundary.

[FNOTFLD]
"buffer not fielded"
The buffer is not a fielded buffer or has not been initialized by Finit().

See Also
Introduction to FML Functions, Falloc, Falloc32(3fml), Frealloc,
Frealloc32(3fml)
free(3), malloc(3) in a UNIX system reference manual

72 ATMI FML Function Reference
Fget, Fget32(3fml)

Name
Fget(), Fget32() - get copy and length of field occurrence

Synopsis
#include <stdio.h>
#include "fml32.h"

int Fget(FBFR *fbfr, FLDID fieldid, FLDOCC oc, char *loc, FLDLEN *maxlen)
int Fget32(FBFR32 *fbfr, FLDID32 fieldid, FLDOCC32 oc, char *loc,
             FLDLEN32 *maxlen)

Description
Fget() should be used to retrieve a field from a fielded buffer when the value is to be modified. fbfr is a pointer to a fielded buffer, fieldid is a field identifier, oc is the occurrence number of the field. The caller provides Fget() with a pointer to a private data area, loc, as well as the length of the data area, *maxlen, and the length of the field is returned in *maxlen. If maxlen is NULL when the function is called, then it is assumed that the data area for the field value loc is big enough to contain the field value and the length of the value is not returned. If loc is NULL, the value is not retrieved. Thus, the function call can be used to determine the existence of the field.

In the “Synopsis” section above the value argument to Fget() is described as a character pointer data type (char * in C). Technically, this describes only one particular kind of value passable to Fget(). In fact, the type of the value argument should be a pointer to an object of the same type as the type of the fielded-buffer representation of the field being retrieved. For example, if the field is stored in the buffer as type FLD_LONG, then value should be of type pointer-to-long (long * in C). Similarly, if the field is stored as FLD_SHORT, then value should be of type pointer-to-short (short * in C). The important thing is that Fget() assumes that the object pointed to by value has the same type as the stored type of the field being retrieved.

Fget32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fget() or Fget32() while running in any context state, including TPINVALIDCONTEXT.

Return Values
When Fget32() is used with the FLD_VIEW32 field type, a pointer to the FVIEWFLD structure is returned. This function returns -1 on error and sets Ferror to indicate the error condition.
Errors Under the following conditions, Fget() fails and sets Ferror to:

[FALIGNERR]
"fielded buffer not aligned"
The buffer does not begin on the proper boundary.

[FNOTFLD]
"buffer not fielded"
The buffer is not a fielded buffer or has not been initialized by Finit().

[FNOSPACE]
"no space"
The size of the data area, as specified in maxlen, is not large enough to hold
the field value.

[FNOTPRES]
"field not present"
A field occurrence is requested but the specified field and/or occurrence was
not found in the fielded buffer.

[FBADFLD]
"unknown field number or type"
A field identifier is specified which is not valid.

See Also Introduction to FML Functions, CFget, CFget32(3fml), Fgetalloc,
Fgetalloc32(3fml), Fgetlast, Fgetlast32(3fml), Fgets, Fgets32(3fml),
Fgetsa, Fgetsa32(3fml)
Fgetalloc, Fgetalloc32(3fml)

Name
Fgetalloc(), Fgetalloc32() - allocate space and get copy of field occurrence

Synopsis
#include <stdio.h>
#include "fml.h"
char *
Fgetalloc(FBFR *fbfr, FLID fieldid, FLDOCC oc, FLDLEN *extralen)
#include "fml32.h"
char *
Fgetalloc32(FBFR32 *fbfr, FLID32 fieldid, FLDOCC32 oc, FLDLEN32 *
extralen)

Description
Like Fget(), Fgetalloc() finds and makes a copy of a buffer field, but it acquires space for the field via a call to malloc() (in UNIX system programmer’s reference manuals). fbfr is a pointer to a fielded buffer. fieldid is a field identifier. oc is the occurrence number of the field. The last argument to Fgetalloc(), extralen, provides an extra amount of space to be acquired in addition to the field value size. It can be used if the retrieved value is to be expanded before reinsertion into the fielded-buffer. If extralen is NULL, then no additional space is allocated and the actual length is not returned. It is the caller’s responsibility to free() space acquired by Fgetalloc(). The buffer will be aligned properly for any field type.

Fgetalloc32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fgetalloc() or Fgetalloc32() while running in any context state, including TPINVALIDCONTEXT.

Return Values
In the “Synopsis” section above the return value to Fgetalloc() is described as a character pointer data type (char * in C). Actually, the pointer returned points to an object that has the same type as the stored type of the field. When Fgetalloc32() is used with the FLD_VIEW32 field type, a pointer to the FVIEWFLD structure is returned.

This function returns NULL on error and sets Ferror to indicate the error condition.

Errors
Under the following conditions, Fgetalloc() fails and sets Ferror to:

[FALIGNERR]
“fielded buffer not aligned”
The buffer does not begin on the proper boundary.
Section 3fml - FML Functions

[FNOTFLD]
"buffer not fielded"
The buffer is not a fielded buffer or has not been initialized by Finit().

[FNOTPRES]
"field not present"
A field occurrence is requested but the specified field and/or occurrence was not found in the fielded buffer.

[FBADFLD]
"unknown field number or type"
A field identifier is specified which is not valid.

[FMALLOC]
“malloc failed”
Allocation of space dynamically using malloc() failed.

See Also
Introduction to FML Functions, CFget, CFget32(3fml), Fget, Fget32(3fml), Fgetlast, Fgetlast32(3fml), Fgets, Fgets32(3fml), Fgetsa, Fgetsa32(3fml) free(3), malloc(3) in a UNIX system reference manual
Fgetlast, Fgetlast32(3fml)

Name
Fgetlast(), Fgetlast32() - get copy of last occurrence

Synopsis
#include <stdio.h>
#include "fml.h"

int
Fgetlast(FBFR *fbfr, FLDID fieldid, FLDOCC *oc, char *value, FLDLEN
*maxlen)

#include "fml32.h"
int
Fgetlast32(FBFR32 *fbfr, FLDID32 fieldid, FLDOCC32 *oc, char
*value, FLDLEN32 *maxlen)

Description
Fgetlast() is used to retrieve both the value and occurrence number of the last
occurrence of the field identified by fieldid. fbfr is a pointer to a fielded buffer. In
order to return the occurrence number of the last field, the occurrence argument, oc, is
a pointer-to-integer, not an integer.

The caller provides Fgetlast() with a pointer to a private buffer, loc, as well as the
length of the buffer, *maxlen, and the length of the field is returned in *maxlen. If
maxlen is NULL when the function is called, then it is assumed that the buffer for the
field value is big enough to contain the field value and the length of the value is not
returned. If loc is NULL, the value is not returned. If oc is NULL, the occurrence is
not returned.

In the “Synopsis” section above the value argument to Fgetlast() is described as a
character pointer data type (char * in C). Technically, this describes only one
particular kind of value passable to Fgetlast(). In fact, the type of the value
argument should be a pointer to an object of the same type as the type of the
fielded-buffer representation of the field being retrieved. For example, if the field is
stored in the buffer as type FLD_LONG, then value should be of type pointer-to-long
(long * in C). Similarly, if the field is stored as FLD_SHORT, then value should be of
type pointer-to-short (short * in C). The important thing is that Fgetlast() assumes
that the object pointed to by value has the same type as the stored type of the field being
retrieved.

Fgetlast32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fgetlast() or
Fgetlast32() while running in any context state, including TPINVALIDCONTEXT.
Return Values  
This function returns -1 on error and sets Ferror to indicate the error condition.

Errors  
Under the following conditions, Fgetlast() fails and sets Ferror to:

[FALIGNERR]  
"fielded buffer not aligned"  
The buffer does not begin on the proper boundary.

[FNOTFLD]  
"buffer not fielded"  
The buffer is not a fielded buffer or has not been initialized by Finit().

[FNOSPACE]  
"no space"  
The size of the data area, as specified in maxlen, is not large enough to hold the field value.

[FNOTPRES]  
"field not present"  
A field occurrence is requested but the specified field and/or occurrence was not found in the fielded buffer.

[FBADFLD]  
"unknown field number or type"  
A field identifier is specified which is not valid.

See Also  
Introduction to FML Functions, Fget, Fget32(3fml), Fgetalloc, Fgetalloc32(3fml), Fgets, Fgets32(3fml), Fgetsa, Fgetsa32(3fml)
Fgets(), Fgets32(3fml)

Name

Fgets(), Fgets32() - get value converted to string

Synopsis

#include <stdio.h>
#include "fml.h"

int
Fgets(FBFR *fbfr, FLDID fieldid, FLDOCC oc, char *buf)
Fgets32(FBFR *fbfr, FLDID fieldid, FLDOCC oc, char *buf)

Description

Fgets() retrieves a field occurrence from the fielded buffer first converting the value
to a user type of FLD_STRING. fbfr is a pointer to a fielded buffer. fieldid is a field
identifier. oc is the occurrence number of the field. The caller of Fgets() provides
buf, a pointer to a private buffer, which is used for the retrieved field value. It is
assumed that buf is large enough to hold the value. Basically, Fgets() calls CFget()
with an assumed utype of FLD_STRING, and a ulen of 0.

Fgets32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fgets() or Fgets32() while running in any context state, including TPINVALIDCONTEXT.

Return Values

This function returns -1 on error and sets Ferror to indicate the error condition.

Errors

Under the following conditions, Fgets() fails and sets Ferror to:

[FALIGNERR]
"fielded buffer not aligned"
The buffer does not begin on the proper boundary.

[FNOTFLD]
"buffer not fielded"
The buffer is not a fielded buffer or has not been initialized by Finit().

[FNOTPRES]
"field not present"
A field occurrence is requested but the specified field and/or occurrence was
not found in the fielded buffer.

ATMI FML Function Reference  79
[FBADFLD]  
"unknown field number or type"  
A field identifier is specified which is not valid.

[FTYPERR]  
"invalid field type"  
A field identifier is specified which is not valid.

[F_MALLOC]  
"malloc failed"  
Allocation of space dynamically using malloc() failed.

See Also  
Introduction to FML Functions, CFget, CFget32(3fml), Fget, Fget32(3fml), Fgetalloc, Fgetalloc32(3fml), Fgetlast, Fgetlast32(3fml), Fgetsa, Fgetsa32(3fml)
### Name
Fgetsa(), Fgetsa32() - use malloc() to allocate space and get converted value

### Synopsis
```c
#include <stdio.h>
#include "fml.h"
char * Fgetsa(FBFR *fbfr, FLID fieldid, FLDOCC oc, FLDLEN *extra)
#include "fml32.h"
char * Fgetsa32(FBFR32 *fbfr, FLID32 fieldid, FLDOCC32 oc, FLDLEN32 *extra)
```

### Description
Fgetsa() is a macro that calls CFgetalloc(). `fbfr` is a pointer to a fielded buffer. `fieldid` is a field identifier. `oc` is the occurrence number of the field. The function uses malloc() (in UNIX system programmer’s reference manuals) to allocate space for the retrieved field value that has been converted to a string. If `extra` is not NULL, it specifies the extra space to allocate in addition to the field value size; the total size is returned in `extra`.

It is the responsibility of the user to free() (in UNIX system reference manuals) the space malloc()'d.

Fgetsa32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fgetsa() or Fgetsa32() while running in any context state, including TPINVALIDCONTEXT.

### Return Values
On success, the function returns a pointer to the allocated buffer.

This function returns NULL on error and sets Ferror to indicate the error condition.

### Errors
Under the following conditions, Fgetsa() fails and sets Ferror to:

[FALIGNERR]
"fielded buffer not aligned"
The buffer does not begin on the proper boundary.

[FNOTFLD]
"buffer not fielded"
The buffer is not a fielded buffer or has not been initialized by Finit().
Section 3fml - FML Functions

[FNOTPRES]  
"field not present"  
A field occurrence is requested but the specified field and/or occurrence was not found in the fielded buffer.

[FBADFLD]  
"unknown field number or type"  
A field identifier is specified which is not valid.

[FTYPERR]  
"invalid field type"  
A field identifier is specified which is not valid.

[Fmalloc]  
"malloc failed"  
Allocation of space dynamically using malloc() failed.

See Also  
Introduction to FML Functions, CFget, CFget32(3fml), Fget, Fget32(3fml), Fgetlast, Fgetlast32(3fml), Fgets, Fgets32(3fml)  
free(3), malloc(3) in a UNIX system reference manual
Fidnm_unload, Fidnm_unload32(3fml)

Name  Fidnm_unload(), Fidnm_unload32() - recover space from id->nm mapping tables

Synopsis  
#include <stdio.h>
#include "fml.h"
void
Fidnm_unload(void);
#include "fml32.h"
void
Fidnm_unload32(void);

Description  Fidnm_unload() recovers space allocated by Fname() for field identifier to field name mapping tables.

Fidnm_unload32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fidnm_unload() or Fidnm_unload32() while running in any context state, including TPINVALIDCONTEXT.

Return Values  This function is declared as a void and so does not return anything.

See Also  Introduction to FML Functions, Fname, Fname32(3fml), Fnmid_unload, Fnmid_unload32(3fml)
Fidxused, Fidxused32(3fml)

Name       Fidxused(), Fidxused32() - return amount of space used

Synopsis   #include <stdio.h>
            #include "fml.h"
            long
            Fidxused(FBFR *fbfr)
            #include "fml32.h"
            long
            Fidxused32(FBFR32 *fbfr)

Description Fidxused() indicates the current amount of space used by the buffer's index. fbfr is a pointer to a fielded buffer.

Fidxused32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fidxused() or Fidxused32() while running in any context state, including TPINVALIDCONTEXT.

Return Values On success, the function returns the amount of space in the buffer used by the index. This function returns -1 on error and sets Ferror to indicate the error condition.

Errors     Under the following conditions, Fidxused() fails and sets Ferror to:

[FALIGNERR]  "fielded buffer not aligned"
             The buffer does not begin on the proper boundary.

[FNOTFLD]    "buffer not fielded"
             The buffer is not a fielded buffer or has not been initialized by Finit().

See Also   Introduction to FML Functions, Findex, Findex32(3fml), Frstrindex,
            Frstrindex32(3fml), Funused, Funused32(3fml), Fused, Fused32(3fml)
Fielded, Fielded32

Fielded, Fielded32(3fml)

Name

Fielded(), Fielded32() - return true if buffer is fielded

Synopsis

#include stdio.h>
#include "fml.h"
int
Fielded(FBFR *fbfr)
#include "fml32.h"
int
Fielded32(FBFR32 *fbfr)

Description

Fielded() is used to test whether the specified buffer is fielded. fbfr is a pointer to a fielded buffer.

Fielded32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fielded() or Fielded32() while running in any context state, including TPINVALIDCONTEXT.

Return Values

Fielded() returns true if the buffer is fielded. It returns false if the buffer is not fielded and does not set Ferror in this case.

See Also

Introduction to FML Functions, Finit, Finit32(3fml), Fneeded, Fneeded32(3fml), Fsizeof, Fsizeof32(3fml)
Findex, Findex32(3fml)

Name
Findex(). Findex32() - index a fielded buffer

Synopsis
#include <stdio.h>
#include "fml.h"
int
Findex(FBFR *fbfr, FLDOCC intvl)
#include "fml32.h"
int
Findex32(FBFR32 *fbfr, FLDOCC32 intvl)

Description
The function Findex() is called explicitly to index a fielded buffer. Fbfr is a pointer
to a fielded buffer. The second parameter, intvl, gives the indexing interval, that is,
the ideal separation of indexed fields. If this argument has value 0, then the buffer’s
current indexing value is used. If the current value itself is 0, the value FSTDINTVL
(defaults to 16) is used. Using an indexing value of 1 will ensure that every field in the
buffer is indexed. The size of the index interval and the amount of space allocated to a
buffer’s index are inversely proportional: the smaller the interval, the more fields are
indexed and thus the larger the amount of space used for indexing.

Findex32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Findex() or Findex32()
while running in any context state, including TPINVALIDCONTEXT.

Return Values
This function returns -1 on error and sets Ferror to indicate the error condition.

Errors
Under the following conditions, Findex() fails and sets Ferror to:

[FALIGNERR]
"fielded buffer not aligned"
The buffer does not begin on the proper boundary.

[FNOTFLD]
"buffer not fielded"
The buffer is not a fielded buffer or has not been initialized by Finit().

[FNOSPACE]
"no space in fielded buffer"
An ENTRY is to be added to the index but there is not enough space
remaining in the buffer.

See Also
Introduction to FML Functions, Fidxused, Fidxused32(3fml), Frstrindex,
Frstrindex32(3fml), Funindex, Funindex32(3fml)
Finit, Finit32(3fml)

Name

Finit(), Finit32() - initialize fielded buffer

Synopsis

#include <stdio.h>
#include "fml.h"

int Finit(FBFR *fbfr, FLDLEN buflen)
#include "fml32.h"

int Finit32(FBFR32 *fbfr, FLDLEN32 buflen)

Description

Finit() can be called to initialize a fielded buffer statically. fbfr is a pointer to a
fielded buffer. buflen is the length of the buffer. The function takes the buffer pointer
and buffer length, and sets up the internal structure for a buffer with no fields. Finit()
can also be used to reinitialize a previously used buffer.

Finit32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Finit() or Finit32()
while running in any context state, including TPINVALIDCONTEXT.

Return Values

This function returns -1 on error and sets Ferror to indicate the error condition.

Errors

Under the following conditions, Finit() fails and sets Ferror to:

[FALIGNERR]

"fielded buffer not aligned"
The buffer does not begin on the proper boundary.

[FNOTFLD]

"buffer not fielded"
The buffer pointer is NULL.

[FNOSPACE]

"no space in fielded buffer"
The buffer size specified is too small for a fielded buffer.

Example

The correct way to reinitialize a buffer to have no fields is:

Finit(fbfr, (FLDLEN)sizeof(fbfr));

See Also

Introduction to FML Functions, Falloc, Falloc32(3fml), Fneeded,
Fneeded32(3fml), Frealloc, Frealloc32(3fml)
Section 3fml - FML Functions

Fjoin, Fjoin32(3fml)

**Name**
Fjoin(), Fjoin32() - join source into destination buffer

**Synopsis**
```
#include stdio.h>
#include "fml.h"
int
Fjoin(FBFR *dest, FBFR *src)
#include "fml32.h"
int
Fjoin32(FBFR32 *dest, FBFR32 *src)
```

**Description**
Fjoin() is used to join two fielded buffers based on matching fieldid/occurrence. dest and src are pointers to the destination and source fielded buffers respectively. For fields that match on fieldid/occurrence, the field value is updated in the destination buffer with the value in the source buffer. Fields in the destination buffer that have no corresponding fieldid/occurrence in the source buffer are deleted. If joining buffers results in the removal of a FLD_PTR field, the memory area referenced by the pointer is not modified or freed.

This function may fail due to lack of space if the new values are larger than the old; in this case, the destination buffer is modified. If this happens, however, the destination buffer may be reallocated using Frealloc() and repeated calls to the Fjoin() function. Even if the destination buffer has been partially updated, the correct results are obtained by repeating the Fjoin() function.

Fjoin32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fjoin() or Fjoin32() while running in any context state, including TPINVALIDCONTEXT.

**Return Values**
This function returns -1 on error and sets Ferror to indicate the error condition.

**Errors**
Under the following conditions, Fjoin() fails and sets Ferror to:

[FALIGNERR]
"fielded buffer not aligned"
Either the source buffer or the destination buffer does not begin on the proper boundary.
Fjoin, Fjoin32(3fml)

[FNOTFLD]
"buffer not fielded"
Either the source buffer or the destination buffer is not a fielded buffer or has not been initialized by Finit().

[FNOSPACE]
"no space in fielded buffer"
A field value is to be added or changed in a field buffer but there is not enough space remaining in the buffer.

Example In the following example:

FBFR *src, *dest; ... if(Fjoin(dest,src) 0) F_error("pgm_name");
if dest has fields A, B, and two occurrences of C, and src has fields A, C, and D, the resultant dest will have source field value A and source field value C.

See Also Introduction to FML Functions, Fconcat, Fconcat32(3fml), Fjoin, Fjoin32(3fml), Fproj, Fproj32(3fml), Fprojcpy, Fprojcpy32(3fml), Frealloc, Frealloc32(3fml)
**Fldid, Fldid32(3fml)**

**Name**  
Fldid(), Fldid32() - map field name to field identifier

**Synopsis**  
#include <stdio.h>  
#include "fml.h"

FLDID  
Fldid(char *name)

#include "fml32.h"

FLDID32  
Fldid32(char *name)

**Description**  
Fldid() provides a run-time translation of a field name to its field identifier and returns a FLDID corresponding to its field name parameter. The first invocation causes space to be dynamically allocated for the field tables and the tables to be loaded. To recover data space used by the field tables loaded by Fldid(), the user may unload the files by a call to the Fnmid_unload() function.

Fldid32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fldid() or Fldid32() while running in any context state, including TPINVALIDCONTEXT.

**Return Values**  
This function returns BADFLDID on error and sets Ferror to indicate the error condition.

**Errors**  
Under the following conditions, Fldid() fails and sets Ferror to:

**[FBADNAME]**  
"unknown field name"  
A field name is specified which cannot be found in the field tables.

**[FMALLOC]**  
"malloc failed"  
Allocation of space dynamically using malloc() failed.

**See Also**  
Introduction to FML Functions, Fldno, Fldno32(3fml), Fname, Fname32(3fml), Fnmid_unload, Fnmid_unload32(3fml)

malloc(3) in a UNIX system reference manual
Fldno, Fldno32(3fml)

Name  Fldno(), Fldno32() - map field identifier to field number

```
#include <stdio.h>
#include "fml.h"

int Fldno(FLDID fieldid)

#include "fml32.h"

long Fldno32(FLDID32 fieldid)
```

Description

Fldno() accepts a field identifier, fieldid, as a parameter and returns the field number contained in the identifier.

Fldno32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fldno() or Fldno32() while running in any context state, including TP_INVALIDCONTEXT.

Return Values

This function returns the field number and does not return an error.

See Also

Introduction to FML Functions, Flid, Flid32(3fml), Fldtype, Fldtype32(3fml)
Section 3fml - FML Functions

**Fldtype, Fldtype32(3fml)**

**Name**  
Fldtype(), Fldtype32() - map field identifier to field type

**Synopsis**  
#include <stdio.h>  
#include "fml.h"

int Fldtype(FLDID fieldid)

#include "fml32.h"

int Fldtype32(FLDID32 fieldid)

**Description**  
Fldtype() accepts a field identifier, fieldid, and returns the field type contained in the identifier (an integer), as defined in fml.h.

Fldtype32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fldtype() or Fldtype32() while running in any context state, including TPINVALIDCONTEXT.

**Return Values**  
This function returns the field type.

**See Also**  
Introduction to FML Functions, Flid, Flid32(3fml), Flidno, Flidno32(3fml)
**Flen, Flen32(3fml)**

**Name**
Flen(), Flen32() - return len of field occurrence in buffer

**Synopsis**
```
#include <stdio.h>
#include "fml.h"

int Flen(FBFR *fbfr, FLDID fieldid, FLDOCC oc)
#include "fml32.h"

long Flen32(FBFR32 *fbfr, FLDID32 fieldid, FLDOCC32 oc)
```

**Description**
Flen() finds the value of the specified field occurrence in the buffer and returns its length. `fbfr` is a pointer to a fielded buffer. `fieldid` is a field identifier. `oc` is the occurrence number of the field.

For values of type `FLD_PTR`, Flen32() returns a fixed length for a pointer field based on `sizeof(char*)`. For values of type `FLD_FML32`, Flen32() returns the value of Fused32() for the length of the nested buffer. For values of type `FLD_VIEW32`, Flen32() returns the length of the view data plus the length of the viewname.

Flen32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Flen() or Flen32() while running in any context state, including TPINVALIDCONTEXT.

**Return Values**
On success, Flen() returns the field length.

This function returns -1 on error and sets Ferror to indicate the error condition.

**Errors**
Under the following conditions, Flen() fails and sets Ferror to:

[FALIGNERR]
"fielded buffer not aligned"
   The buffer does not begin on the proper boundary.

[FNOTFLD]
"buffer not fielded"
   The buffer is not a fielded buffer or has not been initialized by Finit().
[FNOTPRES]  
"field not present"  
A field occurrence is requested but the specified field and/or occurrence was not found in the fielded buffer.

[FBADFLD]  
"unknown field number or type"  
A field identifier is specified which is not valid.

See Also  
Introduction to FML Functions, Fnum, Fnum32(3fml), Fpres, Fpres32(3fml)
Fmkfldid, Fmkfldid32(3fml)

Name
Fmkfldid(), Fmkfldid32() - make a field identifier

#include <stdio.h>
#include "fml.h"

FLDID
Fmkfldid(int type, FLDID num)
#include "fml.h"

FLDID32
Fmkfldid32(int type, FLDID32 num)

Description
Fmkfldid() allows the creation of a valid field identifier from a valid type (as defined in fml.h) and a field number. This is useful for writing an application generator that chooses field numbers sequentially, or for recreating a field identifier.

type is a valid type (an integer; see Flidtype, Flidtype32(3fml)). num is a field number (it should be an unused field number to avoid confusion with existing fields).

Fmkfldid32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fmkfldid() or Fmkfldid32() while running in any context state, including TPINVALIDCONTEXT.

Return Values
This function returns BADFLDID on error and sets Ferror to indicate the error condition.

Errors
Under the following conditions, Fmkfldid() fails and sets Ferror to:

[FBADFLD]
"unknown field number or type"
A field number is specified which is not valid.

[FTYPERR]
"invalid field type"
A field type is specified which is not valid (as defined in fml.h).

See Also
Introduction to FML Functions, Flidtype, Flidtype32(3fml)
Fmove, Fmove32(3fml)

Name  
Fmove(), Fmove32() - move fielded buffer to destination

Synopsis  
# include <stdio.h>
# include "fml.h"

   int
Fmove(char *dest, FBFR *src)

# include "fml32.h"

   int
Fmove32(char *dest, FBFR32 *src)

Description  
Fmove() should be used when copying from a fielded buffer to any type of buffer. dest and src are pointers to the destination buffer and the source fielded buffers respectively.

The difference between Fmove() and Fcpy() is that Fcpy() expects the destination to be a fielded buffer and thus can make sure it is of sufficient size to accommodate the data from the source buffer. Fmove() makes no such check, blindly moving Fsizeof() bytes of data from the source fielded buffer to the target buffer. The destination buffer must be aligned on a short boundary.

For values of type FLD_PTR, Fmove32() transfers the buffer pointer. The application programmer must manage the reallocation and freeing of buffers when the associated pointer is moved.

Fmove32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fmove() or Fmove32() while running in any context state, including TPINVALIDCONTEXT.

Return Values  
This function returns -1 on error and sets Ferror to indicate the error condition.

Errors  
Under the following conditions, Fmove() fails and sets Ferror to:

[FALIGNERR]
"fielded buffer not aligned"

The source or destination buffer does not begin on the proper boundary.
Fmove, Fmove32(3fml)

[FNOTFLD]
"buffer not fielded"
The source buffer is not a fielded buffer or has not been initialized by
Finit().

See Also  Introduction to FML Functions, Fcpy, Fcpy32(3fml), Fsizeof,
Fsizeof32(3fml)
Fname, Fname32(3fml)

Name
Fname(), Fname32() - map field identifier to field name

Synopsis
#include <stdio.h>
#include "fml.h"

c char * Fname(FLDID fieldid)

#include "fml32.h"

c char * Fname32(FLDID32 fieldid)

Description
Fname() provides a run-time translation of a field identifier, fieldid, to its field
name and returns a pointer to a character string containing the name corresponding to
its argument. The first invocation causes space to be dynamically allocated for the field
tables and the tables to be loaded. The table space used by the mapping tables created
by Fname() may be recovered by a call to the function Fidnm_unload().

Fname32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fname() or Fname32()
while running in any context state, including TPINVALIDCONTEXT.

Return Values
This function returns NULL on error and sets Ferror to indicate the error condition.

Errors
Under the following conditions, Fname() fails and sets Ferror to:

[FBADFLD]
"unknown field number or type"
A field number is specified for which a field name cannot be found or is
invalid (0).

[Fmalloc]
"malloc failed"
Allocation of space dynamically using malloc() failed.

See Also
Introduction to FML Functions, Ffprint, Ffprint32(3fml), Fidnm_unload,
Fidnm_unload32(3fml), Fldid, Fldid32(3fml), Fprint, Fprint32(3fml)
Fneeded, Fneeded32(3fml)

Name  
Fneeded(), Fneeded32() - compute size needed for buffer

Synopsis  
#include <stdio.h>
#include "fml.h"

long Fneeded(FLDCC F, FLDLEN V)
#include "fml32.h"

long Fneeded32(FLDCC32 F, FLDLEN32 V)

Description  
Fneeded() is used to determine the space that must be allocated for a fielded buffer. The F argument is the number of fields, and the V argument is the space for all field values, in bytes.

Fneeded32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fneeded() or Fneeded32() while running in any context state, including TPINVALIDCONTEXT.

Return Values  
This function returns -1 on error and sets Ferror to indicate the error condition.

Errors  
Under the following conditions, Fneeded() fails and sets Ferror to:

[FEINVAL]
"invalid argument to function"
One of the arguments to the function invoked was invalid (for example, number of fields is less than 0, V is 0 or total size is greater than 65534).

See Also  
Introduction to FML Functions, Falloc, Falloc32(3fml), Fielded, Fielded32(3fml), Finit, Finit32(3fml), Fsizeof, Fsizeof32(3fml), Funused, Funused32(3fml), Fused, Fused32(3fml)
Section 3fmI - FML Functions

Fnext, Fnext32(3fmI)

Name
Fnext(), Fnext32() - get next field occurrence

Synopsis
#include <stdio.h>
#include "fml.h"

int Fnext(FBFR *fbfr, FLDID *fieldid, FLDOCC *oc, char *value, FLDLEN *len)

#include "fml32.h"

int Fnext32(FBFR32 *fbfr, FLDID32 *fieldid, FLDOCC32 *oc, char *value, FLDLEN32 *len)

Description
Fnext() finds the next field in the buffer after the specified field occurrence. fbfr is a pointer to a fielded buffer. fieldid is a pointer to a field identifier. oc is a pointer to the occurrence number of the field. value is a pointer to the value of the next field. len is the length of the next value.

The field identifier, FIRSTFLDID, should be specified to get the first field in the buffer (for example, on the first call to Fnext()). If value is not NULL, the next field value is copied into value; *len is used to determine if the buffer has enough space allocated to contain the value. The value’s length is returned in *len. If len is NULL when the function is called, it is assumed that there is enough space and the new value length is not returned. If value is NULL, the value is not retrieved and only fieldid and oc are updated. The *fieldid and *oc parameters are respectively set to the next found field and occurrence. If no more fields are found, 0 is returned (end of buffer) and *fieldid, *oc, and *value are left unchanged. Fields are returned in field identifier order.

Although the type of value is char *, the value returned will be of the same type as the next field being retrieved.

When the type of the field to be retrieved is FLD_VIEW32, the value parameter points to a FVIEWFLD structure. The Fnext() function populates the vname and data fields in the structure.

Fnext32() is used with 32-bit FML.
A thread in a multithreaded application may issue a call to \texttt{Fnext()} or \texttt{Fnext32()} while running in any context state, including \texttt{TPINVALIDCONTEXT}.

**Return Values**

\texttt{Fnext()} returns \texttt{1} when the next occurrence is successfully found. It returns \texttt{0} when the end of the buffer is reached.

This function returns \texttt{-1} on error and sets \texttt{Ferror} to indicate the error condition.

**Errors**

Under the following conditions, \texttt{Fnext()} fails and sets \texttt{Ferror} to:

- [FALIGNERR] "fielded buffer not aligned"
  The buffer does not begin on the proper boundary.

- [FNOTFLD] "buffer not fielded"
  The buffer is not a fielded buffer or has not been initialized by \texttt{Finit()}.

- [FNOSPACE] "no space"
  The size of value, as specified in \texttt{len}, is not large enough to hold the field value.

- [FEINVAL] "invalid argument to function"
  One of the arguments to the function invoked was invalid, (for example, specifying NULL for \texttt{fieldid} or \texttt{oc}).

**See Also**

Introduction to FML Functions, \texttt{Fget, Fget32(3fml), Fnum, Fnum32(3fml)}
**Fnmid_unload, Fnmid_unload32(3fml)**

**Name**  
Fnmid_unload(), Fnmid_unload32() - recover space from nm->id mapping tables

**Synopsis**  
#include <stdio.h>  
#include "fml.h"  
void Fnmid_unload(void)  
#include "fml32.h"  
void Fnmid_unload32(void)

**Description**  
To recover data space used by the field tables loaded by Fldid(), the user may unload the files by a call to the Fnmid_unload() function.

Fnmid_unload32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fnmid_unload() or Fnmid_unload32() while running in any context state, including TPINVALIDCONTEXT.

**Return Values**  
This function is declared as a void and so does not return anything.

**See Also**  
Introduction to FML Functions, Fidnm_unload, Fidnm_unload32(3fml), Fldid, Fldid32(3fml)
# Fnum, Fnum32(3fml)

## Name
Fnum(), Fnum32() - return count of all occurrences in buffer

## Synopsis
```
#include <stdio.h>
#include "fml.h"

FLDOCC
Fnum(FBFR *fbfr)
#include "fml32.h"
FLDOCC32
Fnum32(FBFR32 *fbfr)
```

## Description
Fnum() returns the number of fields contained in the specified buffer. `fbfr` is a pointer to a fielded buffer. The `FLD_FML32` and `FLD_VIEW32` fields are each counted as a single field, regardless of the number of fields they contain.

Fnum32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to `Fnum()` or `Fnum32()` while running in any context state, including TPINVALIDCONTEXT.

## Return Values
This function returns -1 on error and sets Ferror to indicate the error condition.

## Errors
Under the following conditions, Fnum() fails and sets Ferror to:

- **[FALIGNERR]**
  "fielded buffer not aligned"
  The buffer does not begin on the proper boundary.

- **[FNOTFLD]**
  "buffer not fielded"
  The buffer is not a fielded buffer or has not been initialized by Finit().

## See Also
Introduction to FML Functions, Foccur, Foccur32(3fml), Fpres, Fpres32(3fml)
**Foccur, Foccur32(3fml)**

**Name**  
Foccur(), Foccur32() - return count of field occurrences in buffer

**Synopsis**  
#include <stdio.h>  
#include "fml.h"

FDLCC  
Foccur(FBFR *fbfr, FLDID fieldid)

#include "fml32.h"

FDLCC32 Foccur32(FBFR32 *fbfr, FLDID32 fieldid)

**Description**  
Foccur() is used to determine the number of occurrences of the field specified by fieldid in the buffer referenced by fbfr. Occurrences of a field within an embedded FML32 buffer are not counted, as in the FLD_FML32 field type.

Foccur32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Foccur() or Foccur32() while running in any context state, including TP_INVALIDCONTEXT.

**Return Values**  
On success, Foccur() returns the number of occurrences; if none are found, it returns 0.

This function returns -1 on error and sets Ferror to indicate the error condition.

**Errors**  
Under the following conditions, Foccur() fails and sets Ferror to:

[FALIGNERR]  
"fielded buffer not aligned"  
The buffer does not begin on the proper boundary.

[FNOTFLD]  
"buffer not fielded"  
The buffer is not a fielded buffer or has not been initialized by Finit().

[FBADFLD]  
"unknown field number or type"  
A field identifier is specified which is not valid.

**See Also**  
Introduction to FML Functions, Fnum, Fnum32(3fml), Fpres, Fpres32(3fml)
Fojoin, Fojoin32(3fml)

Name

Fojoin(), Fojoin32() - outer join source into destination buffer

#include <stdio.h>
#include "fml.h"

int
Fojoin(FBFR *dest, FBFR *src)

#include "fml32.h"

int
Fojoin32(FBFR32 *dest, FBFR32 *src)

Description

Fojoin() is similar to Fjoin(), but it keeps fields from the destination buffer, dest, that have no corresponding fieldid/occurrence in the source buffer, src. Fields that exist in the source buffer that have no corresponding fieldid/occurrence in the destination buffer are not added to the destination buffer. If joining buffers results in the removal of a FLD_PTR field, the memory area referenced by the pointer is not modified or freed.

As with Fjoin(), this function can fail for lack of space; it can be reissued to complete the operation after more space is allocated.

Fojoin32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fojoin() or Fojoin32() while running in any context state, including TPINVALIDCONTEXT.

Return Values

This function returns -1 on error and sets Ferror to indicate the error condition.

Errors

Under the following conditions, Fjoin() fails and sets Ferror to:

[FALIGNERR]
"fielded buffer not aligned"
Either the source buffer or the destination buffer does not begin on the proper boundary.

[FNOTFLD]
"buffer not fielded"
Either the source buffer or the destination buffer is not a fielded buffer or has not been initialized by Finit().
Section 3fml - FML Functions

[FNSPACE]
"no space in fielded buffer"
A field value is to be added or changed in a field buffer but there is not enough
space remaining in the buffer.

Example
In the following example,

if(Fjoin(dest,src) 0)
  F_error("pgm_name");

if dest has fields A, B, and two occurrences of C, and src has fields A, C, and D, the
resultant dest will contain the source field value A, the destination field value B, the
source field value C, and the second destination field value C.

See Also
Introduction to FML Functions, Fconcat, Fconcat32(3fml), Fjoin,
Fjoin32(3fml), Fproj, Fproj32(3fml)
Fpres, Fpres32(3fml)

**Name**
Fpres(), Fpres32() - true if field occurrence is present in buffer

```c
#include <stdio.h>
#include "fml.h"

int Fpres(FBFR *fbfr, FLDID fieldid, FLDOCC oc)
#include "fml32.h"

int Fpres32(FBFR32 *fbfr, FLDID32 fieldid, FLDOCC32 oc)
```

**Description**
Fpres() is used to detect whether a given occurrence (oc) of a specified field (fieldid) exists in the buffer referenced by fbfr. Fpres32() does not check for occurrences of the specified field within an embedded buffer, as in the FLD_FML32 field type.

Fpres32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fpres() or Fpres32() while running in any context state, including TPINVALIDCONTEXT.

**Return Values**
Fpres() returns true if the specified occurrence exists and false otherwise.

**See Also**
Introduction to FML Functions, Ffind, Ffind32(3fml), Fnum, Fnum32(3fml), Foccur, Foccur32(3fml)
## Fprint, Fprint32(3fml)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fprint(), Fprint32()</td>
<td>print buffer to standard output</td>
</tr>
</tbody>
</table>

### Synopsis

```c
#include <stdio.h>
#include "fml.h"

int Fprint(FBFR *fbfr)

#include "fml32.h"

int Fprint32(FBFR32 *fbfr)
```

### Description

Fprint() prints the specified buffer to the standard output. `fbfr` is a pointer to a fielded buffer. For each field in the buffer, the output prints the field name and field value separated by a tab. `Fname()` is used to determine the field name; if the field name cannot be determined, then the field identifier is printed. Non-printable characters in string and character array field values are represented by a backslash followed by their two-character hexadecimal value. A newline is printed following the output of the printed buffer.

For values of type `FLD_PTR`, Fprint32() prints the field name or field identifier and the pointer value in hexadecimal. Although this function prints pointer information, the Fextread32() function ignores the FLD_PTR field type.

For values of type `FLD_FML32`, Fprint32() recursively prints the FML32 buffer, with leading tabs added for each level of nesting. For values of type `FLD_VIEW32`, Fprint32() prints the VIEW32 field name and structure member name/value pairs.

Fprint32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fprint() or Fprint32() while running in any context state, including TPINVALIDCONTEXT.

### Return Values

This function returns -1 on error and sets Ferror to indicate the error condition.

### Errors

Under the following conditions, Fprint() fails and sets Ferror to:

[FALIGNERR]  
"fielded buffer not aligned"

The buffer does not begin on the proper boundary.
[FNOTFLD]
"buffer not fielded"
The buffer is not a fielded buffer or has not been initialized by Finit().

[F_MALLOC]
"malloc failed"
Allocation of space dynamically using malloc() failed.

See Also  Introduction to FML Functions, Fextread, Fextread32(3fml), Ffprint, Ffprint32(3fml), Fname, Fname32(3fml)
Fproj, Fproj32(3fml)

Name
Fproj(), Fproj32() - projection on buffer

Synopsis
#include <stdio.h>
#include "fml.h"

int Fproj(FBFR *fbfr, FLDID *fieldid)

#include "fml32.h"

int Fproj32(FBFR32 *fbfr, FLDID32 *fieldid)

Description
Fproj() is used to update a buffer so as to keep only the desired fields. fbfr is a pointer to a fielded buffer. The desired fields are specified in an array of field identifiers pointed to by fieldid. The last entry in the array must be BADFLDID. The update is done in place; fields that are not in the result of the projection are deleted from the fielded buffer. The array of field identifiers may be rearranged. (If they are not already in numeric order, they are sorted.) If updating buffers results in the removal of a FLD_PTR field, the memory area referenced by the pointer is not modified or freed.

Fproj32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fproj() or Fproj32() while running in any context state, including TPINVALIDCONTEXT.

Return Values
This function returns -1 on error and sets Ferror to indicate the error condition.

Errors
Under the following conditions, Fproj() fails and sets Ferror to:

[FALIGNERR]
"fielded buffer not aligned"
The buffer does not begin on the proper boundary.

[FNOTFLD]
"buffer not fielded"
The buffer is not a fielded buffer or has not been initialized by Finit().

Example
#include "fld.tbl.h"
FBFR *fbfr;
FLDID fieldid[20];
If the buffer has fields A, B, C, and D, the example results in a buffer that contains only occurrences of fields A and D. The entries in the array of field identifiers do not need to be in any specific order, but the last value in the array of field identifiers must be field identifier 0 (BADFLDID).

See Also
Introduction to FML Functions, Fjoin, Fjoin32(3fml), Fojoin, Fojoin32(3fml), Fprojcpy, Fprojcpy32(3fml)
# Section 3fml - FML Functions

## Fprojcpy, Fprojcpy32(3fml)

<table>
<thead>
<tr>
<th>Name</th>
<th>Fprojcpy(), Fprojcpy32() - projection and copy on buffer</th>
</tr>
</thead>
</table>
| Synopsis  | #include <stdio.h>  
            #include "fml.h"  
            int Fprojcpy(FBFR *dest, FBFR *src, FLDID *fieldid)  
            #include "fml32.h"  
            int Fprojcpy32(FBFR32 *dest, FBFR32 *src, FLDID32 *fieldid) |
| Description | Fprojcpy() is similar to Proj() but the projection is done into a destination buffer instead of in-place. dest and src are pointers to the destination and source fielded buffers respectively. fieldid is a pointer to an array of field identifiers. Any fields in the destination buffer are first deleted and the results of the projection on the source buffer are put into the destination buffer. The source buffer is not changed. The array of field identifiers may be rearranged. (If they are not already in numeric order, they are sorted.) If updating buffers results in the removal of a FLD_PTR field, the memory area referenced by the pointer is not modified or freed.  
            This function can fail for lack of space; it can be reissued after allocating enough additional space to complete the operation.  
            Fprojcpy32() is used with 32-bit FML.  
            A thread in a multithreaded application may issue a call to Fprojcpy() or Fprojcpy32() while running in any context state, including TPINVALIDCONTEXT. |
| Return Values | This function returns -1 on error and sets Ferror to indicate the error condition. |
| Errors | Under the following conditions, Fprojcpy() fails and sets Ferror to:  
            [FALIGNERR]  
            "fielded buffer not aligned"  
            Either the source buffer or the destination buffer does not begin on the proper boundary. |

---

112 ATMI FML Function Reference
Fprojcpy, Fprojcpy32(3fml)

[FNOTFLD]  "buffer not fielded"
Either the source buffer or the destination buffer is not a fielded buffer or has
not been initialized by Finit().

[FNOSPACE]  "no space in fielded buffer"
A field value is to be copied to the destination fielded buffer but there is not
enough space remaining in the buffer.

See Also Introduction to FML Functions, Fjoin, Fjoin32(3fml), Fojoin,
Fojoin32(3fml), Fproj, Fproj32(3fml)
### Fread, Fread32(3fml)

**Name**
Fread(), Fread32() - read fielded buffer

**Synopsis**
```c
#include <stdio.h>
#include "fml.h"

int Fread(FBFR *fbfr, FILE *iop)

#include "fml32.h"

int Fread32(FBFR32 *fbfr, FILE32 *iop)
```

**Description**
Fielded buffers may be read from file streams using Fread(). 
`fbfr` is a pointer to a fielded buffer. 
`iop` is a pointer of type FILE to the input stream. 
(See stdio(3S) in a UNIX system reference manual for a discussion of streams). 
Fread() reads the fielded buffer from the stream into fbfr, clearing any previously stored in the buffer, and recreates the buffer’s index. 
Fread32() ignores the FLD_PTR field type. 
No error is returned if a value of type FLD_PTR is supplied to the function.

Fread32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fread() or Fread32() while running in any context state, including TP_INVALIDCONTEXT.

**Return Values**
This function returns -1 on error and sets Ferror to indicate the error condition.

**Errors**
Under the following conditions, Fread() fails and sets Ferror to:

- **[FALIGNERR]**
  "fielded buffer not aligned"
  The buffer does not begin on the proper boundary.

- **[FNOTFLD]**
  "buffer not fielded"
  The buffer is not a fielded buffer or has not been initialized by Finit(). 
  This error is also returned if the data that is read is not a fielded buffer.
[FNOSPACE]
"no space in fielded buffer"
There is not enough space in the buffer to hold the fielded buffer being read from the stream.

[FEUNIX]
"UNIX system call error"
The read() system call failed. The external integer errno should have been set to indicate the error by the system call.

Portability This function is not supported using the BEA Tuxedo System Workstation DLL for Windows.

See Also Introduction to FML Functions, Findex, Findex32(3fml), Fwrite, Fwrite32(3fml)

stdio(3S) in a UNIX system reference manual
Frealloc, Frealloc32(3fml)

Name
Frealloc(), Frealloc32() - reallocate fielded buffer

Synopsis
#include <stdio.h>
#include "fml.h"

FBFR *
Frealloc(FBFR *fbfr, FLDOCC nf, FLDLEN nv)

#include "fml32.h"

FBFR32 *
Frealloc32(FBFR32 *fbfr, FLDOCC32 nf, FLDLEN32 nv)

Description
Frealloc() can be used to reallocate space to enlarge a fielded buffer. fbfr is a pointer to a fielded buffer. The second and third parameters are the new number of fields, nf, and the new number of bytes value space, nv. These are not increments. Frealloc32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Frealloc() or Frealloc32() while running in any context state, including TPINVALIDCONTEXT.

Return Values
On success, Frealloc() returns a pointer to the reallocated FBFR.

This function returns NULL on error and sets Ferror to indicate the error condition.

Errors
Under the following conditions, Frealloc() fails and sets Ferror to:

[FALIGNERR]
"fielded buffer not aligned"
The buffer does not begin on the proper boundary.

[FNOTFLD]
"buffer not fielded"
The buffer is not a fielded buffer or has not been initialized by Finit().

[FEINVAL]
"invalid argument to function"
One of the arguments to the function invoked was invalid (for example, number of fields is less than 0, n is 0 or total size is greater than 65534).
[FMALLOC]  
"malloc failed"  
The new size is smaller than what is currently in the buffer, or allocation of space dynamically using `realloc()` failed.

See Also  
Introduction to FML Functions, `Falloc`, `Falloc32(3fml)`, `Ffree`, `Ffree32(3fml)`
**Frstrindex, Frstrindex32(3fml)**

**Name**  
*Frstrindex(), Frstrindex32* - restore index in a buffer

**Synopsis**  
```c
#include <stdio.h>
#include "fml.h"

int Frstrindex(FBFR *fbfr, FLDOCC numidx)

#include "fml32.h"

int Frstrindex32(FBFR32 *fbfr, FLDOCC32 numidx)
```

**Description**  
A fielded buffer that has been unindexed may be reindexed by either calling *Findex()* or *Frstrindex32*. *fbfr* is a pointer to a fielded buffer. The former performs a total index calculation on the buffer, and is fairly expensive (requiring a full scan of the buffer). It should be used when an unindexed buffer has been altered, or the previous state of the buffer is unknown (for example, when it has been sent from one process to another without an index). *Frstrindex()* is much faster, but may only be used if the buffer has not been altered since its previous unindexing operation. The second argument to *Frstrindex(), numidx, is the return from the *Funindex()* function. *Frstrindex32()* is used with 32-bit FML.

A thread in a multithreaded application may issue a call to *Frstrindex()* or *Frstrindex32()* while running in any context state, including TPINVALIDCONTEXT.

**Return Values**  
This function returns -1 on error and sets *Ferror* to indicate the error condition.

**Errors**  
Under the following conditions, *Frstrindex()* fails and sets *Ferror* to:

[FALIGNERR]  
"fielded buffer not aligned"

The buffer does not begin on the proper boundary.

[FNOTFLD]  
"buffer not fielded"

The buffer is not a fielded buffer or has not been initialized by *Finit()*.
Example

In order to transmit a buffer without its index, something like the following should be performed:

```
save = Funindex(fbfr);
num_to_send = Fused(fbfr);
transmit(fbfr,num_to_send); /* A hypothetical function */
Frstrindx(fbfr,save);
```

These four statements do the following:

1. /* unindex, saving for Frstrindx */
2. /* determine number of bytes to send */
3. /* send fbfr, without index */
4. /* restore index */

In this case, `transmit()` is passed a memory pointer and a length. The data to be transmitted begins at the memory pointer and has `num_to_send` number of significant bytes. Once the buffer has been sent, its index may be restored (assuming `transmit()` does not alter it in any way) using `Frstrindx()`. On the receiving end of the transmission, the process accepting the fielded buffer would index it with `Findex()`, as in:

```
receive(fbfr); /* get fbfr from wherever .. into fbfr */
Findex(fbfr); /* index it */
```

The receiving process cannot call `Frstrindx()` because:

1. It did not call `Funindex()` and so has no idea of what the value of the `numidx` argument to `Frstrindx()` should be.
2. The index itself is not available because it was not sent.

The solution is to call `Findex()` explicitly. Of course, the user is always free to transmit the indexed versions of a fielded buffer (that is, send `Fsizeof(*fbfr)` bytes) and avoid the cost of `Findex()` on the receiving side.

See Also

Introduction to FML Functions, `Findex`, `Findex32(3fml)`, `Fsizeof`, `Fsizeof32(3fml)`, `Funindex`, `Funindex32(3fml)`


Fsizeof, Fsizeof32(3fml)

Name
Fsizeof(), Fsizeof32() - return size of fielded buffer

Synopsis
#include <stdio.h>
#include "fml.h"

long Fsizeof(FBFR *fbfr)
#include "fml32.h"

long Fsizeof32(FBFR32 *fbfr)

Description
Fsizeof() returns the size of a fielded buffer in bytes. fbfr is a pointer to a fielded buffer.

Fsizeof32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fsizeof() or Fsizeof32() while running in any context state, including TPINVALIDCONTEXT.

Return Values
This function returns -1 on error and sets Ferror to indicate the error condition.

Errors
Under the following conditions, Fsizeof() fails and sets Ferror to:

[FALIGNERR]
"fielded buffer not aligned"
The buffer does not begin on the proper boundary.

[FNOTFLD]
"buffer not fielded"
The buffer is not a fielded buffer or has not been initialized by Finit().

See Also
Introduction to FML Functions, Fidxused, Fidxused32(3fml), Fused, Fused32(3fml), Funused, Funused32(3fml)
Fstrerror, Fstrerror32(3fml)

Name   Fstrerror(), Fstrerror32() - get error message string for FML error

Synopsis #include <fml.h>

    char *
    Fstrerror(int err)

#include <fml32.h>

    char *
    Fstrerror32(int err)

Description Fstrerror() is used to retrieve the text of an error message from LIBFML_CAT. err
is the error code set in F_error when a FML function call returns a -1 or other failure
value.

The user can use the pointer returned by Fstrerror() as an argument to userlog() or F_error.

Fstrerror32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fstrerror() or
Fstrerror32() while running in any context state, including TPINVALIDCONTEXT.

Return Values If err is an invalid error code, Fstrerror() returns a NULL. On success, the function
returns a pointer to a string that contains the error message text.

Errors Fstrerror() returns a NULL on error, but does not set F_error.

See Also Introduction to FML Functions, tpstrerror(3c), userlog(3c), F_error,
F_error32(3fml)
**Ftypcvt, Ftypcvt32(3fml)**

**Name**  
Ftypcvt(), Ftypcvt32() - convert from one field type to another

**Synopsis**  
```
#include <stdio.h>
#include "fml.h"

char * Ftypcvt(FLDLEN *tolen, int totype, char *fromval, int fromtype, FLDLEN fromlen)

#include "fml32.h"

char * Ftypcvt32(FLDLEN32 *tolen, int totype, char *fromval, int fromtype, FLDLEN32 fromlen)
```

**Description**  
Ftypcvt() converts the value *fromval*, which has type *fromtype*, and length *fromlen* (if *fromtype* is FLD_CARRAY; otherwise, *fromlen* is inferred from *fromtype*), to a value of type *totype*. Ftypcvt() returns a pointer to the converted value, and sets *tolen* to the converted length, upon success. Upon failure, Ftypcvt() returns NULL.

Ftypcvt32() fails if any of the following field types is used: FLD_PTR, FLD_FML32, or FLD_VIEW32. If one of these field types is encountered, Ferror is set to FEBADOP.

Ftypcvt32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Ftypcvt() or Ftypcvt32() while running in any context state, including TPINVALIDCONTEXT.

**Return Values**  
This function returns NULL on error and sets Ferror to indicate the error condition.

**Errors**  
Under the following conditions, Ftypcvt() fails and sets Ferror to:

```
[F_MALLOC]
"malloc failed"
```

Allocation of space dynamically using malloc() failed when converting from a carray to string.
[FEINVAL]  "invalid argument to function"
    One of the arguments to the function invoked was invalid, (for example, a
    NULL *token or *fromval parameter was specified).

[FTYPERR]  "invalid field type"
    A field identifier is specified which is not valid.

[FTYPERR]  "invalid field type"
    An invalid field type (such as FLD_PTR, FLD_FML32, and FLD_VIEW32) is
    specified.

See Also  Introduction to FML Functions, CFadd, CFadd32(3fml), CFchg, CFchg32(3fml),
          CFfind, CFfind32(3fml), CFget, CFget32(3fml), CFgetalloc,
          CFgetalloc32(3fml)
Ftype, Ftype32(3fml)

Name
Ftype(), Ftype32() - return pointer to type of field

Synopsis
#include <stdio.h>
#include "fml.h"

char * Ftype(FLDID fieldid)

#include "fml32.h"

char * Ftype32(FLDID32 fieldid)

Description
Ftype() returns a pointer to a string containing the name of the type of a field, given a field identifier, fieldid. For example, if the FLDID of a field of type short is supplied to Ftype(), a pointer is returned to the string “short.” This data area is “read-only.”

Ftype32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Ftype() or Ftype32() while running in any context state, including TPINVALIDCONTEXT.

Return Values
On success, Ftype() returns a pointer to a character string that identifies the field type. This function returns NULL on error and sets Ferror to indicate the error condition.

Errors
Under the following conditions, Ftype() fails and sets Ferror to:

[FTYPERR]
  "invalid field type"
  A field identifier is specified which is not valid.

See Also
Introduction to FML Functions, Fldid, Fldid32(3fml), Fldno, Fldno32(3fml)
Funindex, Funindex32(3fml)

**Name**
Funindex(), Funindex32() - discard fielded buffer’s index

**Synopsis**

```c
#include <stdio.h>
#include "fml.h"

FLDOCC
Funindex(FBFR *fbfr)

#include "fml32.h"

FLDOCC32
Funindex32(FBFR32 *fbfr)
```

**Description**

Funindex() discards a fielded buffer’s index. `fbfr` is a pointer to a fielded buffer. When the function returns successfully, the buffer is unindexed. As a result, none of the buffer’s space is allocated to an index and more space is available to user fields (at the cost of potentially slower access time). Unindexing a buffer is useful when it is to be stored on disk or to be transmitted somewhere. In the first case disk space is conserved, in the second, transmission costs may be reduced.

The number of significant bytes from the buffer start, after a buffer has been unindexed is determined by the function call: `Fused(fbfr)`

Funindex32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to `Funindex()` or `Funindex32()` while running in any context state, including `TPINVALIDCONTEXT`.

**Return Values**

Funindex() returns the number of index elements the buffer has before the index is stripped.

This function returns -1 on error and sets `Ferror` to indicate the error condition.

**Errors**

Under the following conditions, `Funindex()` fails and sets `Ferror` to:

[FALIGNERR] "fielded buffer not aligned"
The buffer does not begin on the proper boundary.
[FNOTFLD]
"buffer not fielded"

The buffer is not a fielded buffer or has not been initialized by Finit().

See Also
Introduction to FML Functions, Findex, Findex32(3fml), Frstrindex, Frstrindex32(3fml), Fsizeof, Fsizeof32(3fml), Funused, Funused32(3fml)
Funused, Funused32(3fml)

Name  Funused(), Funused32() - return number of unused bytes in fielded buffer

Synopsis  #include <stdio.h>
          #include "fml.h"

          long
          Funused(FBFR *fbfr)

          #include "fml32.h"

          long
          Funused32(FBFR32 *fbfr)

Description  Funused() returns the amount of space currently unused in the buffer. Space is unused if it contains neither user data nor overhead data such as the header and index.

          fbfr is a pointer to a fielded buffer.

          Funused32() is used with 32-bit FML.

          A thread in a multithreaded application may issue a call to Funused() or
          Funused32() while running in any context state, including TPINVALIDCONTEXT.

Return Values  This function returns -1 on error and sets Ferror to indicate the error condition.

Errors  Under the following conditions, Funused() fails and sets Ferror to:

          [FALIGNERR]
          "fielded buffer not aligned"
          The buffer does not begin on the proper boundary.

          [FNOTFLD]
          "buffer not fielded"
          The buffer is not a fielded buffer or has not been initialized by Finit().

See Also  Introduction to FML Functions, Fidxused, Fidxused32(3fml), Fused,
          Fused32(3fml)
Fupdate, Fupdate32(3fml)

Name  
Fupdate(), Fupdate32() - update destination buffer with source

Synopsis  
#include <stdio.h>
#include "fml.h"

int Fupdate(FBFR *dest, FBFR *src)

#include "fml32.h"

int Fupdate32(FBFR32 *dest, FBFR32 *src)

Description  
Fupdate() updates the destination buffer with the field values in the source buffer. dest and src are pointers to fielded buffers. For fields that match on fieldid/occurrence, the field value is updated in the destination buffer with the value in the source buffer. Fields in the destination buffer that have no corresponding field in the source buffer are left untouched. Fields in the source buffer that have no corresponding field in the destination buffer are added to the destination buffer.

For values of type FLD_PTR, Fupdate32() stores the pointer value. The buffer pointed to by a FLD_PTR field must be allocated using the tpalloc() call. For values of type FLD_FML32, Fupdate32() stores the entire FLD_FML32 field value, except the index. For values of type FLD_VIEW32, Fupdate32() stores a pointer to a structure of type FVIEW32, that contains vflags (a flags field, currently unused and set to 0), vname (a character array containing the viewname), and data (a pointer to the view data stored as a C structure). The application provides the vname and data to Fupdate32().

Fupdate32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fupdate() or Fupdate32() while running in any context state, including TP_INVALIDCONTEXT.

Return Values
This function returns -1 on error and sets Ferror to indicate the error condition.

Errors
Under the following conditions, Fupdate() fails and sets Ferror to:
[FALIGNERR]  
"fielded buffer not aligned"
Either the source buffer or the destination buffer does not begin on the proper boundary.

[FNOTFLD]  
"buffer not fielded"
The source or destination buffer is not a fielded buffer or has not been initialized by Finit().

[FNSPACE]  
"no space in fielded buffer"
A field value is to be added or changed in the destination buffer but there is not enough space remaining in the buffer.

See Also  
Introduction to FML Functions, Fjoin, Fjoin32(3fml), Fofjoin, Fofjoin32(3fml), Fproj, Fproj32(3fml), Fprojcpy, Fprojcpy32(3fml)
### Fused, Fused32(3fml)

**Name**  
Fused(), Fused32() - return number of used bytes in fielded buffer

**Synopsis**  
```c
#include <stdio.h>
#include "fml.h"

long Fused(FBFR *fbfr)

#include "fml32.h"

long Fused32(FBFR32 *fbfr)
```

**Description**  
Fused() returns the amount of used space in a fielded buffer in bytes, including both user data and the header (but not the index, which can be dropped at any time). `fbfr` is a pointer to a fielded buffer.

Fused32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fused() or Fused32() while running in any context state, including TPINVALIDCONTEXT.

**Return Values**  
This function returns -1 on error and sets Ferror to indicate the error condition.

**Errors**  
Under the following conditions, Fused() fails and sets Ferror to:

- **[FALIGNERR]**  
  "fielded buffer not aligned"  
  The buffer does not begin on the proper boundary.

- **[FNOTFLD]**  
  "buffer not fielded"  
  The buffer is not a fielded buffer or has not been initialized by Finit().

**See Also**  
Introduction to FML Functions, Fidxused, Fidxused32(3fml), Funused, Funused32(3fml)
Fvall, Fvall32(3fml)

**Name**
Fvall(), Fvall32() - return long value of field occurrence

```c
#include <stdio.h>
#include "fml.h"

long Fvall(FBFR *fbfr, FLDID fieldid, FLDOCC oc)
#include "fml32.h"

long Fvall32(FBFR32 *fbfr, FLDID fieldid, FLDOCC32 oc)
```

**Description**
Fvall() works like Ffind() for long and short values, but returns the actual value of the field as a long, instead of a pointer to the value. fbfr is a pointer to a fielded buffer. fieldid is a field identifier. oc is the occurrence number of the field.

If the specified field occurrence is not found, then 0 is returned. This function is useful for passing the value of a field to another function without checking the return value. This function is valid only for fields of type FLD_LONG or FLD_SHORT.

Fvall32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fvall() or Fvall32() while running in any context state, including TPINVALIDCONTEXT.

**Return Values**
For fields of types other than FLD_LONG or FLD_SHORT, Fvall() returns 0 and sets Ferror to FTYPERR.

This function returns 0 on other errors and sets Ferror to indicate the error condition.

**Errors**
Under the following conditions, Fvall() fails and sets Ferror to:

[FALIGNERR]
"fielded buffer not aligned"
The buffer does not begin on the proper boundary.

[FNOTFLD]
"buffer not fielded"
The buffer is not a fielded buffer or has not been initialized by Finit().
Section 3fml - FML Functions

[FBADFLD]
"unknown field number or type"
A field identifier is specified which is not valid.

[FTYPERR]
"invalid field type"
Bad fieldid or the field type is not FLD_SHORT or FLD_LONG.

See Also  Introduction to FML Functions, Ffind, Ffind32(3fml), Fvals, Fvals32(3fml)
# Fvals, Fvals32(3fml)

**Name**  
Fvals(), Fvals32() - return string value of field occurrence

**Synopsis**  
```c
#include <stdio.h>
#include "fml.h"

char *
Fvals(FBFR *fbfr, FLDID fieldid, FLDOCC oc)

#include "fml32.h"

char *
Fvals32(FBFR32 *fbfr, FLDOCC32 fieldid, FLDOCC32 oc)
```

**Description**  
Fvals() works like Ffind() for string values but guarantees that a value is returned.  
*fbfr* is a pointer to a fielded buffer.  *fieldid* is a field identifier.  *oc* is the occurrence number of the field.

If the specified field occurrence is not found, then the NULL string is returned.  This function is useful for passing the value of a field to another function without checking the return value.  This function is valid only for fields of type FLD_STRING; the NULL string is automatically returned for other field types (that is, no conversion is done).

Fvals32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fvals() or Fvals32() while running in any context state, including TPINVALIDCONTEXT.

**Return Values**  
This function returns the NULL string on error and sets Ferror to indicate the error condition.

**Errors**  
Under the following conditions, Fvals() fails and sets Ferror to:

[FALIGNERR]  
"fielded buffer not aligned"  
The buffer does not begin on the proper boundary.

[FNOTFLD]  
"buffer not fielded"  
The buffer is not a fielded buffer or has not been initialized by Finit().
Section 3fml - FML Functions

[FBADFLD]
"unknown field number or type"
A field identifier is specified which is not valid.

[FTYPERR]
"invalid field type"
Bad fieldid or the field type is not FLD_STRING.

See Also Introduction to FML Functions, CFFind, CFFind32(3fml), Ffind, Ffind32(3fml), Fvall, Fvall32(3fml)
**Fvftos, Fvftos32(3fml)**

**Name**  
Fvftos(), Fvftos32() - copy from fielded buffer to C structure

**Synopsis**  
```c
#include <stdio.h>
#include "fml.h"

int Fvftos(FBFR *fbfr, char *cstruct, char *view)

#include "fml32.h"

int Fvftos32(FBFR32 *fbfr, char *cstruct, char *view)
```

**Description**  
The Fvftos() function transfers data from a fielded buffer to a C structure. `fbfr` is a pointer to a fielded buffer. `cstruct` is a pointer to a C structure. `view` is a pointer to the name of a compiled view description.

Fields are copied from the fielded buffer into the structure based on the member descriptions in the `view`. If a field in the fielded buffer has no corresponding member in the C structure, it is ignored. If a member specified in the C structure has no corresponding field in the fielded buffer, a NULL value is copied into the member. The NULL value used is definable for each member in the view description.

To store multiple occurrences in the C structure, the structure member should be an array (for example, `int zip[4]` can store 4 occurrences of `zip`). If the buffer has fewer occurrences of the field than there are elements in the array, the extra element slots are assigned NULL values. On the other hand, if the buffer has more occurrences of the field than there are elements in the array, the surplus occurrences are ignored.

There are view description options that inhibit mappings even though a mapping entry exists for a field identifier and a member. These options are initially specified in the viewfile, but can be changed at run time using `Fvopt()`.

Fvftos32() is used with 32-bit FML.

A thread in a multithreaded application may issue a call to Fvftos() or Fvftos32() while running in any context state, including TPINVALIDCONTEXT.

**Return Values**  
This function returns -1 on error and sets `Ferror` to indicate the error condition.
Errors Under the following conditions, Fvftos() fails and sets Ferror to:

[FALIGNERR]
"fielded buffer not aligned"
The buffer does not begin on the proper boundary.

[FNOTFLD]
"buffer not fielded"
The buffer is not a fielded buffer or has not been initialized by Finit().

[FEINVAL]
"invalid argument to function"
One of the arguments to the function invoked was invalid, (for example, specifying a NULL cstruct parameter to Fvftos).

[FBADACM]
"ACM contains negative value"
An Associated Count Member should not be a negative value while transferring data from a structure to a fielded buffer.

[FBADVIEW]
"cannot find or get view"
The view description specified was NULL or was not found in the files specified by VIEWDIR or VIEWFILES.

See Also Introduction to FML Functions, Fvopt, Fvopt32(3fml), viewfile(5)
# Fvneeded, Fvneeded32(3fml)

## Name
Fvneeded(), Fvneeded32() - computes size needed for VIEW buffer

## Synopsis
```c
#include <stdio.h>
#include "fml.h"

long Fvneeded(char *subtype)

#include "fml32.h"

long Fvneeded32(char *subtype)
```

## Description
Fvneeded() returns the size of the VIEW C structure. *subtype* is the name of the VIEW. You can call Fvneeded() to determine the size of a VIEW buffer to allocate. Fvneeded32() is used with 32-bit VIEWS.

## Return Values
Fvneeded() returns the size of the VIEW in number of bytes. This function returns -1 on error and sets Ferror to indicate the error condition.

## Errors
Under the following conditions, Fvnull() fails and sets Ferror to:

- [FEINVAL]
  "invalid argument to function"
  The requested VIEW cannot be found in the viewfiles specified by VIEWDIR and VIEWFILES environment variables.

## See Also
Introduction to FML Functions, viewfile(5)
Section 3fml - FML Functions

Fvnull, Fvnull32(3fml)

Name
Fvnull(), Fvnull32() - check if a structure element is NULL

Synopsis
#include <stdio.h>
#include "fml.h"

int Fvnull(char *cstruct, char *cname, FLOCC oc, char *view)

#include "fml32.h"

int Fvnull32(char *cstruct, char *cname, FLOCC32 oc, char *view)

Description
Fvnull() is used to determine if an occurrence of a structure element is NULL.
cstruct is a pointer to a C structure. cname is a pointer to the name of an element
within cstruct. oc is the occurrence number of the element. view is a pointer to the
name of a compiled view description.

Options of Fvopt() such as do not affect this function.

Fvnull32() is used for views defined with viewc32 or VIEW32 typed buffers for
larger views with more fields.

A thread in a multithreaded application may issue a call to Fvnull() or Fvnull32()
while running in any context state, including TPINVALIDCONTEXT.

Return Values
Fvnull() returns 1, if the specified cname in a C structure is NULL and returns 0 if
not NULL. This function returns -1 on error and sets Ferror to indicate the error
condition.

Errors
Under the following conditions, Fvnull() fails and sets Ferror to:

[FBADVIEW]
"cannot find or get view"
The view description specified was not found in the files specified by
VIEWDIR or VIEWFILES.

[FNOCNAME]
"cname not found"
The C structure field name is not found in the view description.
See Also Introduction to FML Functions, Fvopt, Fvopt32(3fml), viewfile(5)
**Fvopt, Fvopt32(3fml)**

**Name**  
Fvopt(), Fvopt32() - change flag options of a mapping entry

**Synopsis**  
#include <stdio.h>  
#include "fml.h"

```c
int Fvopt(char *cname, int option, char *view)
```

#include "fml32.h"

```c
int Fvopt32(char *cname, int option, char *view)
```

**Description**  
Fvopt() allows users to specify buffer-to-structure mapping options at run time.  
`cname` is a pointer to the name of an element in a view description, `view`, `option` specifies the desired setting for the mapping option. Valid options and their meanings are:

- **F_FTOS**  
  One-way mapping from fielded buffer to structure, flag `S` in the view description.

- **F_STOF**  
  One-way mapping from structure to fielded buffer, flag `F` in the view description.

- **F_OFF**  
  No mapping between the fielded buffer and the structure, flag `N` in the view description.

- **F_BOTH**  
  Two-way mapping between the fielded buffer and the structure, flag `S, F` in the view description.

Fvopt32() is used for views defined with viewc32 or VIEW32 typed buffers for larger views with more fields.

A thread in a multithreaded application may issue a call to Fvopt() or Fvopt32() while running in any context state, including TPINVALIDCONTEXT.

**Return Values**  
This function returns -1 on error and sets Ferror to indicate the error condition.
Errors

Under the following conditions, Fvopt() fails and sets Ferror to:

[FEINVAL]
"invalid argument to function"
One of the arguments to the function invoked was invalid (for example, specifying a NULL cname or view parameter or specifying an invalid option).

[FBADVIEW]
"cannot find or get view"
The view was not found in the files specified by VIEWDIR and VIEWFILES.

[FNOCNAME]
"cname not found"
The C structure field name is not found in the view description.

See Also

Introduction to FML Functions, viewfile(5)
## Fvrefresh, Fvrefresh32(3fml)

<table>
<thead>
<tr>
<th>Name</th>
<th>Fvrefresh(), Fvrefresh32() - copy from C structure to fielded buffer</th>
</tr>
</thead>
</table>
| Synopsis | #include <stdio.h>  
#include "fml.h"  

void Fvrefresh()  
#include "fml32.h"  

void Fvrefresh32() |
| Description | Fvrefresh() clears and reinitializes the internal cache of view structure mappings. This is necessary only when frequently accessed views are updated dynamically. Fvrefresh32() is used for views defined with viewc32 or VIEW32 typed buffers for larger views with more fields. A thread in a multithreaded application may issue a call to Fvrefresh() or Fvrefresh32() while running in any context state, including TPINVALIDCONTEXT. |
| Return Values | This routine is a void function and does not return a value. |
| Errors | This routine is a void function and no error codes are set. |
| See Also | Introduction to FML Functions |
Fvselinit, Fvselinit32(3fml)

Name
Fvselinit(), Fvselinit32() - initialize structure element to NULL

Synopsis
#include <stdio.h>
#include "fml.h"

int Fvselinit(char *cstruct, char *cname, char *view)

#include "fml32.h"

int Fvselinit32(char *cstruct, char *cname, char *view)

Description
Fvselinit() initializes an individual element of a C structure to its appropriate NULL value. cstruct is a pointer to a C structure. cname is a pointer to the name of an element of cstruct. view is a pointer to the name of a compiled view description.

Fvselinit() sets the associated count member of the element to 0 if the c flag was used when the view was compiled, and sets the associated length member to the length of the associated NULL value if the l flag was used in the viewfile.

Fvselinit32() is used for views defined with viewc32 or VIEW32 typed buffers for larger views with more fields.

A thread in a multithreaded application may issue a call to Fvselinit() or Fvselinit32() while running in any context state, including TPINVALIDCONTEXT.

Return Values
This function returns -1 on error and sets Ferror to indicate the error condition.

Errors
Under the following conditions, Fvselinit() fails and sets Ferror to:

[FEINVAL]
"invalid argument to function"
One of the arguments to the function invoked was invalid (for example, specifying a NULL cstruct parameter invalid Fvselinit).

[FBADVVIEW]
"cannot find or get view"
The view description specified was NULL or was not found in the files specified by VIEWDIR or VIEWFILES.
Section 3fml - FML Functions

[FNOCNAME]
"cname not found"
The C structure field name is not found in the view description.

See Also  Introduction to FML Functions, Fvsnit, Fvsnit32(3fml), viewfile(5)
**Fvsinit, Fvsinit32(3fml)**

**Name**  
Fvsinit(), Fvsinit32() - initialize C structure to NULL

**Synopsis**  
#include <stdio.h>  
#include "fml.h"

    int
    Fvsinit(char *cstruct, char *view)

#include "fml32.h"

    int
    Fvsinit32(char *cstruct, char *view)

**Description**  
Fvsinit() initializes all members in a C structure to the NULL values specified in the view description. view. cstruct is a pointer to a C structure. view is a pointer to a compiled view description.

Fvsinit() sets the associated count member of an element to 0 if the C flag was used when the view was compiled, and sets the associated length member to the length of the associated NULL value if the L flag was used in the viewfile.

Fvsinit32() is used for views defined with viewc32 or VIEW32 typed buffers for larger views with more fields.

A thread in a multithreaded application may issue a call to Fvsinit() or Fvsinit32() while running in any context state, including TPINVALIDCONTEXT.

**Return Values**  
This function returns -1 on error and sets Ferror to indicate the error condition.

**Errors**  
Under the following conditions, Fvsinit() fails and sets Ferror to:

    [FEINVAL]  
    "invalid argument to function"  
    One of the arguments to the function invoked was invalid (for example, specifying a NULL cstruct parameter invalid Fvsinit()).

    [FBADVVIEW]  
    "cannot find or get view"  
    The view description specified was NULL or was not found in the files specified by VIEWDIR or VIEWFILES.

**See Also**  
Introduction to FML Functions, Fvselinit, Fvselinit32(3fml), viewfile(5)
### Fvstof, Fvstof32(3fml)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fvstof(), Fvstof32()</td>
<td>copy from C structure to fielded buffer</td>
</tr>
</tbody>
</table>

#### Synopsis

```c
#include <stdio.h>
#include "fml.h"

int Fvstof(FBFR *fbfr, char *cstruct, int mode, char *view)

#include "fml32.h"

int Fvstof32(FBFR32 *fbfr, char *cstruct, int mode, char *view)
```

#### Description

Fvstof() transfers data from a C structure to a fielded buffer. `fbfr` is a pointer to a fielded buffer. `cstruct` is a pointer to a C structure. `mode` specifies the manner in which the transfer is made. `view` is a pointer to a compiled view description. `mode` has four possible values:

- **FUPDATE**
- **FOJOIN**
- **FJOIN**
- **FCONCAT**

The action of these modes are the same as that described in `Fupdate()`, `Fojoin()`, `Fjoin()`, and `Fconcat()`. One can even think of `Fvstof()` as the same as these functions, except that where they specify a source buffer, `Fvstof()` specifies a C structure. Bear in mind that `FUPDATE` does not move structure elements that have NULL values.

`Fvstof32()` is used for views defined with `viewc32` or `VIEW32` typed buffers for larger views with more fields.

A thread in a multithreaded application may issue a call to `Fvstof()` or `Fvstof32()` while running in any context state, including `TPINVALIDCONTEXT`.

#### Return Values

This function returns -1 on error and sets `Ferror` to indicate the error condition.
Errors

Under the following conditions, Fvstof() fails and sets Ferror to:

[FALIGNERR]  
"fielded buffer not aligned"  
The buffer does not begin on the proper boundary.

[FNOTFLD]  
"buffer not fielded"  
The buffer is not a fielded buffer or has not been initialized by Finit().

[FEINVAL]  
"invalid argument to function"  
One of the arguments to the function invoked was invalid (for example,  
specifying a NULL cstruct parameter or an invalid mode to Fvstof())

[FNOSPACE]  
"no space in fielded buffer"  
A field value is to be added or changed in a fielded buffer but there is not  
enough space remaining in the buffer.

[FBADACM]  
"ACM contains negative value"  
An Associated Count Member should not be a negative value while  
transferring data from a structure to a fielded buffer.

[FMALLOC]  
"malloc failed"  
Allocation of space dynamically using malloc() failed when converting  
from a carray or string value.

See Also
Introduction to FML Functions, Fconcat, Fconcat32(3fml), Fjoin,  
Fjoin32(3fml), Fojoin, Fojoin32(3fml), Fupdate, Fupdate32(3fml),  
Fvftos, Fvftos32(3fml)
**Fvstot, Fvttos (3fml)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Fvstot(), Fvttos() - convert C structure to/from target record type</th>
</tr>
</thead>
</table>
| Synopsis| #include <stdio.h>  
#include "fml.h"  
long Fvstot(char *cstruct, char *trecord, long treclen, char *viewname)  
long Fvttos(char *cstruct, char *trecord, char *viewname)  
#include "fml32.h"  
int Fvstot32(char *cstruct, char *trecord, long treclen, char *viewname)  
int Fvttos32(char *cstruct, char *trecord, char *viewname)  
int Fcodeset(char *translation_table) |
| Description| The Fvstot() function transfers data from a C structure to a target record type. The Fvttos() function transfers data from a target record to a C structure. trec is a pointer to the target record. cstruct is a pointer to a C structure. viewname is a pointer to the name of a compiled view description. The VIEWDIR and VIEWFILES are used to find the directory and file containing the compiled view description. Fvttos32() and Fvstot32() are used with 32-bit VIEWS.  
To convert from an FML buffer to a target record, first call Fvttos() to convert the FML buffer to a C structure, and call Fvstot() to convert to a target record. To convert from a target record to an FML buffer, first call Fvttos() to convert to a C structure and then call Fvstof() to convert the structure to an FML buffer.  
A thread in a multithreaded application may issue a call to Fvstot() or Fvttos() while running in any context state, including TPINVALIDCONTEXT. |
The default target is IBM/370 COBOL records. The default data conversion is done based on the following table.

### Table 2: Default Data Conversion

<table>
<thead>
<tr>
<th>Struct</th>
<th>Record</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>COMP-1</td>
</tr>
<tr>
<td>double</td>
<td>COMP-2</td>
</tr>
<tr>
<td>long</td>
<td>S9(9) COMP</td>
</tr>
<tr>
<td>short</td>
<td>S9(4) COMP</td>
</tr>
<tr>
<td>int</td>
<td>S9(9) COMP or S9(4) COMP</td>
</tr>
<tr>
<td>dec_t(m, n)</td>
<td>S9(2*m-(n+1))V9(n)COMP-3</td>
</tr>
<tr>
<td>ASCII char</td>
<td>EBCDIC char</td>
</tr>
<tr>
<td>ASCII string</td>
<td>EBCDIC string</td>
</tr>
<tr>
<td>carray</td>
<td>character array</td>
</tr>
</tbody>
</table>

No filler bytes are provided between fields in the IBM/370 record. The COBOL SYNC clause should not be specified for any data items that are a part of the structure corresponding to the view.

An integer field is converted to either a four or two-byte integer depending on the size of integers on the machine on which the conversion is done.

A string field in the view must be terminated with a NULL when converting to/from the IBM/370 format.

The data in a carray field is passed unchanged; no data translation is performed.

Packed decimals exist in the IBM/370 environment as two decimal digits packed into one byte with the low-order half byte used to store the sign. The length of a packed decimal may be 1 to 16 bytes with storage available for 1 to 31 digits and a sign.
Packed decimals are supported in C structures using the `dec_t` field type. The `dec_t` field has a defined size consisting of two numbers separated by a comma. The number to the left of the comma is the total number of bytes that the decimal occupies. The number to the right is the number of digits to the right of the decimal point. The formula for conversion is:

$$\text{dec}_t(m, n) = \text{S9}(2^m - (n+1)) \text{V9}(n) \text{COMP}-3$$

Decimal values may be converted to and from other data types (for example, int, long, string, double, and float) using the functions described in `decimal()`.

The following table provides the hex values for default character conversion of ASCII (on the left) to/from EBCDIC (on the right).

<table>
<thead>
<tr>
<th>00 00</th>
<th>01 01</th>
<th>02 02</th>
<th>03 03</th>
<th>04 04</th>
<th>05 06</th>
<th>06 07</th>
<th>07 08</th>
</tr>
</thead>
<tbody>
<tr>
<td>08 09</td>
<td>0a 0b</td>
<td>0c 0d</td>
<td>0e 0f</td>
<td>10 11</td>
<td>12 13</td>
<td>14 15</td>
<td>16 17</td>
</tr>
<tr>
<td>18 19</td>
<td>1a 1b</td>
<td>1c 1d</td>
<td>1e 1f</td>
<td>20 21</td>
<td>22 23</td>
<td>24 25</td>
<td>26 27</td>
</tr>
<tr>
<td>28 29</td>
<td>2a 2b</td>
<td>2c 2d</td>
<td>2e 2f</td>
<td>30 31</td>
<td>32 33</td>
<td>34 35</td>
<td>36 37</td>
</tr>
<tr>
<td>38 39</td>
<td>3a 3b</td>
<td>3c 3d</td>
<td>3e 3f</td>
<td>40 41</td>
<td>42 43</td>
<td>44 45</td>
<td>46 47</td>
</tr>
<tr>
<td>48 49</td>
<td>4a 4b</td>
<td>4c 4d</td>
<td>4e 4f</td>
<td>50 51</td>
<td>52 53</td>
<td>54 55</td>
<td>56 57</td>
</tr>
<tr>
<td>58 59</td>
<td>5a 5b</td>
<td>5c 5d</td>
<td>5e 5f</td>
<td>60 61</td>
<td>62 63</td>
<td>64 65</td>
<td>66 67</td>
</tr>
<tr>
<td>68 69</td>
<td>6a 6b</td>
<td>6c 6d</td>
<td>6e 6f</td>
<td>70 71</td>
<td>72 73</td>
<td>74 75</td>
<td>76 77</td>
</tr>
<tr>
<td>78 79</td>
<td>7a 7b</td>
<td>7c 7d</td>
<td>7e 7f</td>
<td>80 81</td>
<td>82 83</td>
<td>84 85</td>
<td>86 87</td>
</tr>
<tr>
<td>88 89</td>
<td>8a 8b</td>
<td>8c 8d</td>
<td>8e 8f</td>
<td>90 91</td>
<td>92 93</td>
<td>94 95</td>
<td>96 97</td>
</tr>
<tr>
<td>98 99</td>
<td>9a 9b</td>
<td>9c 9d</td>
<td>9e 9f</td>
<td>a0 a1</td>
<td>a2 a3</td>
<td>a4 a5</td>
<td>a6 a7</td>
</tr>
<tr>
<td>a8 a9</td>
<td>aaa ab</td>
<td>ac ac</td>
<td>ad af</td>
<td>b0 b1</td>
<td>b2 b3</td>
<td>b4 b5</td>
<td>b6 b7</td>
</tr>
<tr>
<td>b8 b9</td>
<td>baa bb</td>
<td>bcc bc</td>
<td>bdd df</td>
<td>e0 e1</td>
<td>e2 e3</td>
<td>e4 e5</td>
<td>e6 e7</td>
</tr>
<tr>
<td>e8 e9</td>
<td>ea eb</td>
<td>ec ed</td>
<td>ee ef</td>
<td>f0 f1</td>
<td>f2 f3</td>
<td>f4 f5</td>
<td>f6 f7</td>
</tr>
<tr>
<td>f8 f9</td>
<td>fa fb</td>
<td>fc fd</td>
<td>fe ff</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
An alternate character translation table can be used at run time by calling \texttt{Fcodeset()}. The \texttt{translation_table} must point to 512 bytes of binary data. The first 256 bytes of data are interpreted as the ASCII to EBCDIC translation table. The second 256 bytes of data are interpreted as the EBCDIC to ASCII table. Any data after the 512th byte is ignored. If the pointer is NULL, the default translation is used.

Return Values

On success, \texttt{Fvstot()} returns the length of the target record and \texttt{Fvttos()} returns the length of the C structure.

These functions return -1 on error and set \texttt{Ferror} to indicate the error condition.

Errors

Under the following conditions, \texttt{Fvttos()} fails and sets \texttt{Ferror} to:

**[FEINVAL]**
"invalid argument to function"
One of the arguments to the function invoked was invalid (for example, specifying a NULL \texttt{trecord} or \texttt{cstruct} parameter to \texttt{Fvttos()}). This error is also returned if a value is out of range when converting to or from a target record.

**[FBADACM]**
"ACM contains negative value"
An Associated Count Member cannot be a negative value.

**[FBADVIEW]**
"cannot find or get view"
\texttt{viewname} was not found in the files specified by \texttt{VIEWDIR} or \texttt{VIEWFILES}.

**[FNOSPACE]**
"no space in buffer"
The target record is not large enough to hold the converted structure.

**[FVFOPEN]**
"cannot find or open viewfile"
While trying to find \texttt{viewname}, the program failed to find one of the files specified by \texttt{VIEWDIR} or \texttt{VIEWFILES}.

**[FEUNIX]**
"operating system error"
While trying to find \texttt{viewname}, the program failed to open one of the files specified by \texttt{VIEWDIR} or \texttt{VIEWFILES} for reading.
Section 3FML - FML Functions

[FVFSYNTAX]
"bad viewfile"
While trying to find \texttt{viewname}, one of the files specified by \texttt{VIEWDIR} or \texttt{VIEWFILES} was corrupted or not a viewfile.

[FMALLOC]
"malloc failed"
While trying to find \texttt{viewname}, \texttt{malloc()} failed while allocating space to hold the view information.

Example VIEW test.v

\begin{verbatim}
VIEW test
#type cname fbname count flag size null
float float1 FLOAT1 1 - - 0.0
double double1 DOUBLE1 1 - - 0.0
long long1 LONG1 1 - - 0
short short1 SHORT1 1 - - 0
int int1 INT1 1 - - 0
dec_t dec1 DEC1 1 - 4,2 0
char char1 CHAR1 1 - - ''
string string1 STRING1 1 - 20 ''
carray carray1 CARRAY1 1 - 20 ''
END
\end{verbatim}

Equivalent COBOL Record

\begin{verbatim}
02 OUTPUT-REC.
  05 FLOAT1                USAGE IS COMP-1.
  05 DOUBLE1               USAGE IS COMP-2.
  05 LONG1                 PIC S9(9) USAGE IS COMP.
  05 SHORT1                PIC S9(4) USAGE IS COMP.
  05 INT1                  PIC S9(9) USAGE IS COMP.
  05 DEC1                  PIC S9(5)V9(2) COMP-3.
  05 CHAR1                 PIC X(01).
  05 STRING1               PIC X(20).
  05 CARRAY1               PIC X(20).
\end{verbatim}

C Program

\begin{verbatim}
#include "test.h"
#include "decimal.h"

main()
{

    struct test s1;
\end{verbatim}
char data[100];

s1.float1 = 1.0;
s1.double1 = 2.0;
s1.long1 = 3;
s1.short1 = 4;
s1.int1 = 5;
decvdbl(6.0, sl.decl);
sl.char1 = '7';
(void) strcpy(sl.string1, "eight");
(void) strcpy(sl.carray1, "nine");

if (Fvstot((char *)&s1, data, reclen, "test") == -1) {
    printf("Fvstot failed: %s", Fstrerror(Ferror));
    exit(0);
}
/* transfer to target machine and get response */
...

/* translate back */
if (Fvttos(data, (char *)&s1, "test") == -1) {
    printf("Fvttos failed: %s", Fstrerror(Ferror));
    exit(0);
}
/* use the structure */
......
exit(0);
}

See Also
Introduction to FML Functions, Fvftos, Fvftos32(3fml), Fvstof,
Fvstof32(3fml), viewfile(5)

decimal(3) in a UNIX system reference manual
Fwrite, Fwrite32(3fml)

Name         Fwrite(), Fwrite32() - write fielded buffer
Synopsis     #include <stdio.h>
             #include "fml.h"

             int Fwrite(FBFR *fbfr, FILE *iop)

             #include "fml32.h"

             int Fwrite32(FBFR32 *fbfr, FILE *iop)

Description  Fielded buffers may be written to streams by Fwrite(). (See stdio(3S) in a UNIX
             system reference manual for a discussion of streams). Fwrite() discards a buffer’s
             index.

             fbfr is a pointer to a fielded buffer. iop is a pointer of type FILE to the output stream.

             For the FLD_PTR field type, only the pointer, not the data being pointed to, is written
             to the output stream. For the FLD_VIEW32 field type, only the FVIEWFLD structure, not
             the data in the VIEW32 buffer, is written to the output stream.

             Fwrite32() is used with 32-bit FML.

             A thread in a multithreaded application may issue a call to Fwrite() or Fwrite32()
             while running in any context state, including TPINVALIDCONTEXT.

Return Values This function returns -1 on error and sets Ferror to indicate the error condition.

Errors       Under the following conditions, Fwrite() fails and sets Ferror to:
             [FALIGNERR]
             "fielded buffer not aligned"
             The buffer does not begin on the proper boundary.

             [FNOTFLD]
             "buffer not fielded"
             The buffer is not a fielded buffer or has not been initialized by Finit().
Fwrite, Fwrite32(3fml)

[FEUNIX]
"UNIX system call error"
The write system call failed. The external integer errno should have been set to indicate the error by the system call, and the external integer Uunixerr (values defined in Uunix.h) is set to the system call that returned the error.

Portability  This function is not supported using the BEA Tuxedo System Workstation DLL for Windows.

See Also  Introduction to FML Functions, Findex, Findex32(3fml), Fread, Fread32(3fml)

stdio(3S) in a UNIX system reference manual