open-hea Documentation

Release 0.1

Tiwonge Manda, Brown Msiska, Evidence for Development

September 19, 2013

CONTENTS

Contents:

CHAPTER

ONE

CONTROLLER PACKAGE

1.1 `controller` Package

This file is part of open-hea.

open-hea is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

open-hea is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with open-hea. If not, see <http://www.gnu.org/licenses/>.

CHAPTER

TWO

MODEL PACKAGE

2.1 model Package

This file is part of open-hea.

open-hea is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

open-hea is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with open-hea. If not, see <http://www.gnu.org/licenses/>.

2.2 common Module

This file is part of open-hea.

open-ihm is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

open-ihm is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with open-ihm. If not, see <http://www.gnu.org/licenses/>.

model.common.**getDbString**(*strSeed*)

model.common.**getIntMonth**(*month*)

model.common.**getStringMonth**(*month*)

model.common.**getViewString**(*strSeed*)

2.3 config Module

This file is part of open-ihm.

-- coding: utf-8 --

open-ihm is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

open-ihm is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with open-ihm. If not, see <http://www.gnu.org/licenses/>.

class model.config.**Config**

Bases: object

Configures the mysql database connector which is then called as below:

includes.mysql.connector.Connect(**Config.dbinfo())

CHARSET = 'utf8'

DATABASE = 'openheadb'

HOST = 'localhost'

PASSWORD = 'hea2012'

PORT = 3306

UNICODE = True

USER = 'openhea'

WARNINGS = True

classmethod dbinfo()

2.4 config_parser Module

This file is part of open-hea.

open-hea is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

open-hea is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with open-hea. If not, see <http://www.gnu.org/licenses/>.

class model.config_parser.**OpenHEAConfig**

Bases: ConfigParser.SafeConfigParser

Configuration file parser based on ConfigParser.SafeConfigParser

The parser is setup with defaults for most of the settings.

config = OpenHEAConfig() config.read(CONFIGFILE)

The config file can contain these directives,

[database] host = host database = database user = user password = userpassword port = 3306 superuser = root superuser_password = password driver = mysql+mysqldb

database_config()
Returns a dictionary containing the database configuration information. If it can not load the settings from the ini file it simply returns the default settings.

{ 'host': 'localhost', 'database': 'openheadb', 'user': 'openhea', 'password': 'hea2012', 'port': 3306, 'superuser': 'root', 'superuser_password': '',

}

dbinfo()
Returns a dictionary containing the database configuration information.

{ 'host': 'localhost', 'database': 'openheadb', 'user': 'openhea', 'password': 'hea2012', 'charset': 'utf8', 'use_unicode' : True, 'get_warnings' : True,

}

get_driver()
Returns the database driver to use in sqlalchemy

schema_script_path()
Returns the path to look for the database schema scripts. This is primarily configurable to make testing easier.

sqlalchemy_connection_string()
Returns a connection string for sqlalchemy. The defaults will produce the following connection string,

mysql+mysqldb://openhea:hea2012@localhost/openheadb

sqlalchemy_mysql_unix_socket()
Returns the unix socket specified in the configuration file for the sqlalchemy connection.

This shouldn't be a commonly necessary function, but it is used on our CI server. The parameter should be unix_socket in the configuration file. For example,

unix_socket = /var/run/mysqld/mysqld.sock

sqlalchemy_superuser_connection_string()
Returns a connection string for the superuser for sqlalchemy. The defaults will produce the following connection string,

Note: this may not work with blank passwords. Then again it might. I have no way to test it, your mileage may vary.

superuser_dbinfo()
Returns a dictionary containing the database configuration information for the super user.

{ 'host': 'localhost', 'database': 'openheadb', 'user': 'root', 'password': '', 'charset': 'utf8', 'use_unicode' : True, 'get_warnings' : True,

}

2.5 databaseinitialiser Module

2.6 schema Module

2.7 unstableschema Module

CHAPTER

THREE

VIEW PACKAGE

3.1 `view` Package

This file is part of open-hea.

open-hea is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

open-hea is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with open-hea. If not, see <http://www.gnu.org/licenses/>.

3.2 `frmdatabasemessage` Module

3.3 `frmmainwindow` Module

3.4 `mixins` Module

CHAPTER

FOUR

EXTERNAL MODULES PACKAGED WITH OPEN-HEA (USED FOR READING / WRITING EXCEL SPREADSHEETS) :

4.1 xlrd Package

4.1.1 xlrd Package

class xlrd.__init__.**Book**

Bases: xlrd.biffh.BaseObject

biff2_8_load(*filename=None*, *file_contents=None*, *logfile=<open file '<stdout>', mode 'w' at 0x7fd8360cf150>*, *verbosity=0*, *pickleable=True*, *use_mmap=1*, *encoding_override=None*, *formatting_info=False*, *on_demand=False*)

biff_version = 0

codepage = None

colour_map = {}

countries = (0, 0)

datemode = 0

derive_encoding()

encoding = None

fake_globals_get_sheet()

font_list = []

format_list = []

format_map = {}

get2bytes()

get_record_parts()

get_record_parts_conditional(*reqd_record*)

get_sheet(*sh_number*, *update_pos=True*)

get_sheets()

getbof(*rqd_stream*)

handle_boundsheet(*data*)

handle_builtinfmtcount(*data*)

handle_codepage(*data*)

handle_country(*data*)

handle_datemode(*data*)

handle_externname(*data*)

handle_externsheet(*data*)

handle_filepass(*data*)

handle_name(*data*)

handle_obj(*data*)

handle_sheethdr(*data*)

handle_sheetsoffset(*data*)

handle_sst(*data*)

handle_supbook(*data*)

handle_writeaccess(*data*)

initialise_format_info()

load_time_stage_1 = **-1.0**

load_time_stage_2 = **-1.0**

name_and_scope_map = {}

name_map = {}

name_obj_list = []

names_epilogue()

nsheets = **0**

palette_record = []

parse_globals()

read(*pos*, *length*)

release_resources()

sheet_by_index(*sheetx*)

sheet_by_name(*sheet_name*)

sheet_loaded(*sheet_name_or_index*)

sheet_names()

sheets()

style_name_map = {}

unload_sheet(*sheet_name_or_index*)

user_name = **u'‘**

xf_list = []

class xlrd.__init__.**Name**
Bases: xlrd.biffh.BaseObject

area2d(*clipped=True*)

binary = 0

book = None

builtin = 0

cell()

complex = 0

func = 0

funcgroup = 0

hidden = 0

macro = 0

name = u''

name_index = 0

raw_formula = ''

result = None

scope = -1

vbasic = 0

xlrd.__init__.**colname**(*colx*, *_A2Z='ABCDEFGHIJKLMNOPQRSTUVWXYZ'*)

xlrd.__init__.**count_records**(*filename*, *outfile=<open file '<stdout>', mode 'w' at 0x7fd8360cf150>*)

xlrd.__init__.**display_cell_address**(*rowx*, *colx*, *relrow*, *relcol*)

xlrd.__init__.**dump**(*filename*, *outfile=<open file '<stdout>', mode 'w' at 0x7fd8360cf150>*, *unnumbered=False*)

xlrd.__init__.**expand_cell_address**(*inrow*, *incol*)

xlrd.__init__.**is_cell_opcode**()
D.has_key(k) -> True if D has a key k, else False

xlrd.__init__.**open_workbook**(*filename=None*, *logfile=<open file '<stdout>', mode 'w' at 0x7fd8360cf150>*, *verbosity=0*, *pickleable=True*, *use_mmap=1*, *file_contents=None*, *encoding_override=None*, *formatting_info=False*, *on_demand=False*)

xlrd.__init__.**unpack_SST_table**(*datatab*, *nstrings*)
Return list of strings

4.1.2 biffh Module

class xlrd.biffh.**BaseObject**
Bases: object

dump(*f=None*, *header=None*, *footer=None*, *indent=0*)

exception xlrd.biffh.**XLRDError**
Bases: exceptions.Exception

xlrd.biffh.**biff_count_records**(*mem*, *stream_offset*, *stream_len*, *fout=<open file '<stdout>', mode 'w' at 0x7fd8360cf150>*)

xlrd.biffh.**biff_dump**(*mem*, *stream_offset*, *stream_len*, *base=0*, *fout=<open file '<stdout>', mode 'w' at 0x7fd8360cf150>*, *unnumbered=False*)

xlrd.biffh.**fprintf**(*f*, *fmt*, **vargs*)

xlrd.biffh.**hex_char_dump**(*strg*, *ofs*, *dlen*, *base=0*, *fout=<open file '<stdout>', mode 'w' at 0x7fd8360cf150>*, *unnumbered=False*)

xlrd.biffh.**is_cell_opcode**()
D.has_key(k) -> True if D has a key k, else False

xlrd.biffh.**unpack_cell_range_address_list_update_pos**(*output_list*, *data*, *pos*, *biff_version*, *addr_size=6*)

xlrd.biffh.**unpack_string**(*data*, *pos*, *encoding*, *lenlen=1*)

xlrd.biffh.**unpack_string_update_pos**(*data*, *pos*, *encoding*, *lenlen=1*, *known_len=None*)

xlrd.biffh.**unpack_unicode**(*data*, *pos*, *lenlen=2*)
Return unicode_strg

xlrd.biffh.**unpack_unicode_update_pos**(*data*, *pos*, *lenlen=2*, *known_len=None*)
Return (unicode_strg, updated value of pos)

xlrd.biffh.**upkbits**(*tgt_obj*, *src*, *manifest*, *local_setattr=<built-in function setattr>*)

xlrd.biffh.**upkbitsL**(*tgt_obj*, *src*, *manifest*, *local_setattr=<built-in function setattr>*, *local_int=<type 'int'>*)

4.1.3 compdoc Module

class xlrd.compdoc.**CompDoc**(*mem*, *logfile=<open file '<stdout>', mode 'w' at 0x7fd8360cf150>*, *DEBUG=0*)
Bases: object

get_named_stream(*qname*)

locate_named_stream(*qname*)

exception xlrd.compdoc.**CompDocError**
Bases: exceptions.Exception

class xlrd.compdoc.**DirNode**(*DID*, *dent*, *DEBUG=0*)
Bases: object

dump(*DEBUG=1*)

4.1.4 formatting Module

class xlrd.formatting.**EqNeAttrs**
Bases: object

class xlrd.formatting.**Font**
Bases: xlrd.biffh.BaseObject, xlrd.formatting.EqNeAttrs

bold = 0

character_set = 0

colour_index = 0

escapement = 0

family = 0

font_index = 0

height = 0

italic = 0

name = u''

outline = 0

shadow = 0

struck_out = 0

underline_type = 0

underlined = 0

weight = 400

class xlrd.formatting.**Format**(*format_key*, *ty*, *format_str*)
Bases: xlrd.biffh.BaseObject, xlrd.formatting.EqNeAttrs

format_key = 0

format_str = u''

type = 0

class xlrd.formatting.**XF**
Bases: xlrd.biffh.BaseObject

alignment = None

background = None

border = None

font_index = 0

format_key = 0

is_style = 0

parent_style_index = 0

protection = None

xf_index = 0

class xlrd.formatting.**XFAlignment**
Bases: xlrd.biffh.BaseObject, xlrd.formatting.EqNeAttrs

hor_align = 0

indent_level = 0

rotation = 0

shrink_to_fit = 0

text_direction = 0

text_wrapped = 0

vert_align = 0

class xlrd.formatting.**XFBackground**
Bases: xlrd.biffh.BaseObject, xlrd.formatting.EqNeAttrs

background_colour_index = 0

fill_pattern = 0

pattern_colour_index = 0

class xlrd.formatting.**XFBorder**
Bases: xlrd.biffh.BaseObject, xlrd.formatting.EqNeAttrs

bottom_colour_index = 0

bottom_line_style = 0

diag_colour_index = 0

diag_down = 0

diag_line_style = 0

diag_up = 0

left_colour_index = 0

left_line_style = 0

right_colour_index = 0

right_line_style = 0

top_colour_index = 0

top_line_style = 0

class xlrd.formatting.**XFProtection**
Bases: xlrd.biffh.BaseObject, xlrd.formatting.EqNeAttrs

cell_locked = 0

formula_hidden = 0

xlrd.formatting.**check_colour_indexes_in_obj**(*book*, *obj*, *orig_index*)

xlrd.formatting.**default_palette = {80: ((0, 0, 0), (255, 255, 255), (255, 0, 0), (0, 255, 0), (0, 0, 255), (255, 255, 0), (255,**
00H = Normal 01H = RowLevel_lv (see next field) 02H = ColLevel_lv (see next field) 03H = Comma 04H = Currency 05H = Percent 06H = Comma [0] (BIFF4-BIFF8) 07H = Currency [0] (BIFF4-BIFF8) 08H = Hyperlink (BIFF8) 09H = Followed Hyperlink (BIFF8)

xlrd.formatting.**fmt_bracketed_sub**()
sub(repl, string[, count = 0]) –> newstring Return the string obtained by replacing the leftmost non-overlapping occurrences of pattern in string by the replacement repl.

xlrd.formatting.**handle_efont**(*book*, *data*)

xlrd.formatting.**handle_font**(*book*, *data*)

xlrd.formatting.**handle_format**(*self*, *data*, *rectype=1054*)

xlrd.formatting.**handle_palette**(*book*, *data*)

xlrd.formatting.**handle_style**(*book*, *data*)

xlrd.formatting.**handle_xf**(*self*, *data*)

xlrd.formatting.**initialise_book**(*book*)

xlrd.formatting.**initialise_colour_map**(*book*)

xlrd.formatting.**is_date_format_string**(*book*, *fmt*)

xlrd.formatting.**nearest_colour_index**(*colour_map*, *rgb*, *debug=0*)

xlrd.formatting.**palette_epilogue**(*book*)

xlrd.formatting.**xf_epilogue**(*self*)

4.1.5 formula Module

xlrd.formula.**decompile_formula**(*bk*, *fmla*, *fmlalen*, *reldelta*, *browx=None*, *bcolx=None*, *blah=0*, *level=0*)

xlrd.formula.**dump_formula**(*bk*, *data*, *fmlalen*, *bv*, *reldelta*, *blah=0*, *isname=0*)

xlrd.formula.**evaluate_name_formula**(*bk*, *nobj*, *namex*, *blah=0*, *level=0*)

xlrd.formula.**rangename3d**(*book*, *ref3d*)
Ref3D(1, 4, 5, 20, 7, 10) => 'Sheet2:Sheet3!$H$6:$J$20' (assuming Excel's default sheetnames)

xlrd.formula.**rangename3drel**(*book*, *ref3d*)

xlrd.formula.**cellname**(*rowx*, *colx*)
(5, 7) => 'H6'

xlrd.formula.**cellnameabs**(*rowx*, *colx*)
(5, 7) => '$H$6'

xlrd.formula.**colname**(*colx*)
7 => 'H', 27 => 'AB'

4.1.6 licences Module

Portions copyright © 2005-2009, Stephen John Machin, Lingfo Pty Ltd All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. None of the names of Stephen John Machin, Lingfo Pty Ltd and any contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

4.1.7 sheet Module

class xlrd.sheet.**Cell**(*ctype*, *value*, *xf_index=None*)
Bases: xlrd.biffh.BaseObject

ctype

value

xf_index

class xlrd.sheet.**Colinfo**
Bases: xlrd.biffh.BaseObject

bit1_flag = 0

collapsed = 0

hidden = 0

outline_level = 0

width = 0

xf_index = -1

class xlrd.sheet.**MSNote**
Bases: xlrd.biffh.BaseObject

class xlrd.sheet.**MSODrawing**
Bases: xlrd.biffh.BaseObject

class xlrd.sheet.**MSObj**
Bases: xlrd.biffh.BaseObject

class xlrd.sheet.**MSTxo**
Bases: xlrd.biffh.BaseObject

class xlrd.sheet.**Rowinfo**
Bases: xlrd.biffh.BaseObject

additional_space_above = 0

additional_space_below = 0

has_default_height = 0

has_default_xf_index = 0

height = 0

height_mismatch = 0

hidden = 0

outline_group_starts_ends = 0

outline_level = 0

xf_index = -9999

class xlrd.sheet.**Sheet**(*book*, *position*, *name*, *number*)
Bases: xlrd.biffh.BaseObject

cell(*rowx*, *colx*)

cell_type(*rowx*, *colx*)

cell_value (*rowx*, *colx*)

cell_xf_index (*rowx*, *colx*)

col (*colx*, *start_rowx=0*, *end_rowx=None*)

col_label_ranges = []

col_slice (*colx*, *start_rowx=0*, *end_rowx=None*)

col_types (*colx*, *start_rowx=0*, *end_rowx=None*)

col_values (*colx*, *start_rowx=0*, *end_rowx=None*)

colinfo_map = {}

computed_column_width (*colx*)

default_additional_space_above = None

default_additional_space_below = None

default_row_height = None

default_row_height_mismatch = None

default_row_hidden = None

defcolwidth = None

extend_cells (*nr*, *nc*)

fake_XF_from_BIFF20_cell_attr (*cell_attr*)

fix_ragged_rows ()

fixed_BIFF2_xfindex (*cell_attr*, *rowx*, *colx*, *true_xfx=None*)

gcw = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

handle_feat11 (*data*)

handle_msodrawingetc (*recid*, *data_len*, *data*)

handle_note (*data*)

handle_obj (*data*)

handle_txo (*data*)

merged_cells = []

name = ''

ncols = 0

nrows = 0

put_blank_cell (*rowx*, *colx*, *xf_index*)

put_cell (*rowx*, *colx*, *ctype*, *value*, *xf_index*)

put_number_cell (*rowx*, *colx*, *value*, *xf_index*)

read (*bk*)

req_fmt_info ()

row (*rowx*)

row_label_ranges = []

row_slice(*rowx*, *start_colx=0*, *end_colx=None*)

row_types(*rowx*, *start_colx=0*, *end_colx=None*)

row_values(*rowx*, *start_colx=0*, *end_colx=None*)

rowinfo_map = {}

standardwidth = None

tidy_dimensions()

visibility = 0

xlrd.sheet.**is_cell_opcode**()
D.has_key(k) -> True if D has a key k, else False

xlrd.sheet.**unpack_RK**(*rk_str*)

4.1.8 timemachine Module

xlrd.timemachine.**int_floor_div**(*x*, *y*)

xlrd.timemachine.**intbool**(*x*)

4.1.9 xldate Module

exception xlrd.xldate.**XLDateAmbiguous**
Bases: xlrd.xldate.XLDateError

exception xlrd.xldate.**XLDateBadDatemode**
Bases: xlrd.xldate.XLDateError

exception xlrd.xldate.**XLDateBadTuple**
Bases: xlrd.xldate.XLDateError

exception xlrd.xldate.**XLDateError**
Bases: exceptions.ValueError

exception xlrd.xldate.**XLDateNegative**
Bases: xlrd.xldate.XLDateError

exception xlrd.xldate.**XLDateTooLarge**
Bases: xlrd.xldate.XLDateError

xlrd.xldate.**xldate_as_tuple**(*xldate*, *datemode*)

xlrd.xldate.**xldate_from_date_tuple**(*(year*, *month*, *day)*, *datemode*)

xlrd.xldate.**xldate_from_datetime_tuple**(*datetime_tuple*, *datemode*)

xlrd.xldate.**xldate_from_time_tuple**(*(hour*, *minute*, *second)*)

4.2 xlwt Package

4.2.1 xlwt Package

4.2.2 BIFFRecords Module

class xlwt.BIFFRecords.**BackupRecord**(*backup*)
Bases: xlwt.BIFFRecords.BiffRecord

This record contains a Boolean value determining whether Excel makes a backup of the file while saving.

class xlwt.BIFFRecords.**Biff8BOFRecord**(*rec_type*)
Bases: xlwt.BIFFRecords.BiffRecord

Offset Size Contents 0 2 Version, contains 0600H for BIFF8 and BIFF8X 2 2 Type of the following data:

0005H = Workbook globals 0006H = Visual Basic module 0010H = Worksheet 0020H = Chart 0040H = Macro sheet 0100H = Workspace file

4 2 Build identifier 6 2 Build year 8 4 File history flags 12 4 Lowest Excel version that can read all records in this file

BOOK_GLOBAL = 5

CHART = 32

MACROSHEET = 64

VB_MODULE = 6

WORKSHEET = 16

WORKSPACE = 256

class xlwt.BIFFRecords.**BiffRecord**
Bases: object

get()

get_rec_header()

get_rec_id()

class xlwt.BIFFRecords.**BlankRecord**(*row*, *col*, *xf_index*)
Bases: xlwt.BIFFRecords.BiffRecord

This record represents an empty cell.

Record BLANK, BIFF5-BIFF8:

Offset Size Contents 0 2 Index to row 2 2 Index to first column (fc) 4 2 indexes to XF record

class xlwt.BIFFRecords.**BookBoolRecord**
Bases: xlwt.BIFFRecords.BiffRecord

This record contains a Boolean value determining whether to save values linked from external workbooks (CRN records and XCT records). In BIFF3 and BIFF4 this option is stored in the WSBOOL record.

Record BOOKBOOL, BIFF5-BIFF8:

Offset Size Contents 0 2 0 = Save external linked values;

1 = Do not save external linked values

class xlwt.BIFFRecords.**BoolErrRecord**(*row*, *col*, *xf_index*, *number*, *is_error*)
Bases: xlwt.BIFFRecords.BiffRecord

This record represents a cell that contains a boolean or error value.

class xlwt.BIFFRecords.**BottomMarginRecord**(*margin*)
Bases: xlwt.BIFFRecords.BiffRecord

This record is part of the Page Settings Block. It contains the bottom page margin of the current worksheet.

Offset Size Contents 0 8 Bottom page margin in inches

(IEEE 754 floating-point value, 64?bit double precision)

class xlwt.BIFFRecords.**BoundSheetRecord**(*stream_pos*, *visibility*, *sheetname*, *encoding='ascii'*)
Bases: xlwt.BIFFRecords.BiffRecord

This record is located in the workbook globals area and represents a sheet inside of the workbook. For each sheet a BOUNDSHEET record is written. It stores the sheet name and a stream offset to the BOF record within the workbook stream. The record is also known as BUNDLESHEET.

Record BOUNDSHEET, BIFF5-BIFF8: Offset Size Contents 0 4 Absolute stream position of the BOF record of the sheet represented by this record. This

field is never encrypted in protected files.

4 1 Visibility: 00H = Visible 01H = Hidden 02H = Strong hidden

5 1 Sheet type: 00H = Worksheet 02H = Chart 06H = Visual Basic module

6 var. Sheet name: BIFF5/BIFF7: Byte string, 8-bit string length BIFF8: Unicode string, 8-bit string length

class xlwt.BIFFRecords.**CalcCountRecord**(*calc_count*)
Bases: xlwt.BIFFRecords.BiffRecord

This record is part of the Calculation Settings Block. It specifies the maximum number of times the formulas should be iteratively calculated. This is a fail-safe against mutually recursive formulas locking up a spreadsheet application.

Record CALCCOUNT, BIFF2-BIFF8:

Offset Size Contents 0 2 Maximum number of iterations allowed in circular references

class xlwt.BIFFRecords.**CalcModeRecord**(*calc_mode*)
Bases: xlwt.BIFFRecords.BiffRecord

This record is part of the Calculation Settings Block. It specifies whether to calculate formulas manually, automatically or automatically except for multiple table operations.

Record CALCMODE, BIFF2-BIFF8:

Offset Size Contents 0 2 FFFFH = automatic except for multiple table operations

0000H = manually 0001H = automatically (default)

class xlwt.BIFFRecords.**CodepageBiff8Record**
Bases: xlwt.BIFFRecords.BiffRecord

This record stores the text encoding used to write byte strings, stored as MS Windows code page identifier. The CODEPAGE record in BIFF8 always contains the code page 1200 (UTF-16). Therefore it is not possible to obtain the encoding used for a protection password (it is not UTF-16).

Record CODEPAGE, BIFF2-BIFF8:

Offset Size Contents 0 2 Code page identifier used for byte string text encoding:

016FH = 367 = ASCII 01B5H = 437 = IBM PC CP-437 (US) 02D0H = 720 = IBM PC CP-720 (OEM Arabic) 02E1H = 737 = IBM PC CP-737 (Greek) 0307H = 775 = IBM PC CP-775 (Baltic) 0352H = 850 = IBM PC CP-850 (Latin I) 0354H = 852 = IBM PC CP-852 (Latin II (Central European)) 0357H = 855 = IBM PC CP-855 (Cyrillic) 0359H = 857 = IBM PC CP-857 (Turkish) 035AH = 858 = IBM PC CP-858 (Multilingual Latin I with Euro) 035CH = 860 = IBM PC CP-860 (Portuguese) 035DH = 861 = IBM PC CP-861 (Icelandic) 035EH = 862 = IBM PC CP-862 (Hebrew) 035FH = 863 = IBM PC CP-863 (Canadian (French)) 0360H = 864 = IBM PC CP-864 (Arabic) 0361H = 865 = IBM PC CP-865 (Nordic) 0362H = 866 = IBM PC CP-866 (Cyrillic (Russian)) 0365H = 869 = IBM PC CP-869 (Greek (Modern)) 036AH = 874 = Windows CP-874 (Thai) 03A4H = 932 = Windows CP-932 (Japanese Shift-JIS) 03A8H = 936 = Windows CP-936 (Chinese Simplified GBK) 03B5H = 949 = Windows CP-949 (Korean (Wansung)) 03B6H = 950 = Windows CP-950 (Chinese Traditional BIG5) 04B0H = 1200 = UTF-16 (BIFF8) 04E2H = 1250 = Windows CP-1250 (Latin II) (Central European) 04E3H = 1251 = Windows CP-1251 (Cyrillic) 04E4H = 1252 = Windows CP-1252 (Latin I) (BIFF4-BIFF7) 04E5H = 1253 = Windows CP-1253 (Greek) 04E6H = 1254 = Windows CP-1254 (Turkish) 04E7H = 1255 = Windows CP-1255 (Hebrew) 04E8H = 1256 = Windows CP-1256 (Arabic) 04E9H = 1257 = Windows CP-1257 (Baltic) 04EAH = 1258 = Windows CP-1258 (Vietnamese) 0551H = 1361 = Windows CP-1361 (Korean (Johab)) 2710H = 10000 = Apple Roman 8000H = 32768 = Apple Roman 8001H = 32769 = Windows CP-1252 (Latin I) (BIFF2-BIFF3)

UTF_16 = 1200

class xlwt.BIFFRecords.**ColInfoRecord**(*first_col*, *last_col*, *width*, *xf_index*, *options*)
Bases: xlwt.BIFFRecords.BiffRecord

This record specifies the width for a given range of columns. If a column does not have a corresponding COLINFO record, the width specified in the record STANDARDWIDTH is used. If this record is also not present, the contents of the record DEFCOLWIDTH is used instead. This record also specifies a default XF record to use for cells in the columns that are not described by any cell record (which contain the XF index for that cell). Additionally, the option flags field contains hidden, outline, and collapsed options applied at the columns.

Record COLINFO, BIFF3-BIFF8:

Offset Size Contents 0 2 Index to first column in the range 2 2 Index to last column in the range 4 2 Width of the columns in 1/256 of the width of the zero character, using default font

(first FONT record in the file)

6 2 Index to XF record for default column formatting 8 2 Option flags:

Bits Mask Contents 0 0001H 1 = Columns are hidden 10-8 0700H Outline level of the columns (0 = no outline) 12 1000H 1 = Columns are collapsed

10 2 Not used

class xlwt.BIFFRecords.**ContinueRecord**
Bases: xlwt.BIFFRecords.BiffRecord

Whenever the content of a record exceeds the given limits (see table), the record must be split. Several CONTINUE records containing the additional data are added after the parent record.

BIFF version Maximum data size of a record BIFF2-BIFF7 2080 bytes (2084 bytes including record header) BIFF8 8224 bytes (8228 bytes including record header) (0x2020)

Record CONTINUE, BIFF2-BIFF8: Offset Size Contents 0 var. Data continuation of the previous record

Unicode strings are split in a special way. At the beginning of each CONTINUE record the option flags byte is repeated. Only the character size flag will be set in this flags byte, the Rich-Text flag and the Far-East flag are set to zero. In each CONTINUE record it is possible that the character size changes from 8-bit characters to 16-bit characters and vice versa.

Never a Unicode string is split until and including the first character. That means, all header fields (string length, option flags, optional Rich-Text size, and optional Far-East data size) and the first character of the string have to occur together in the leading record, or have to be moved completely into the CONTINUE record. Formatting runs cannot be split between their components (character index and FONT record index). If a string is split between two formatting runs, the option flags field will not be repeated in the CONTINUE record.

class xlwt.BIFFRecords.**CountryRecord**(*ui_id*, *sys_settings_id*)
Bases: xlwt.BIFFRecords.BiffRecord

This record stores two Windows country identifiers. The first represents the user interface language of the Excel version that has saved the file, and the second represents the system regional settings at the time the file was saved.

Record COUNTRY, BIFF3-BIFF8:

Offset Size Contents 0 2 Windows country identifier of the user interface language of Excel 2 2 Windows country identifier of the system regional settings

The following table shows most of the used country identifiers. Most of these identifiers are equal to the international country calling codes.

1 USA 2 Canada 7 Russia

class xlwt.BIFFRecords.**DSFRecord**
Bases: xlwt.BIFFRecords.BiffRecord

This record specifies if the file contains an additional BIFF5/BIFF7 workbook stream. Record DSF, BIFF8: Offset Size Contents 0 2 0 = Only the BIFF8 Workbook stream is present

1 = Additional BIFF5/BIFF7 Book stream is in the file

A double stream file can be read by Excel 5.0 and Excel 95, and still contains all new features added to BIFF8 (which are left out in the BIFF5/BIFF7 Book stream).

class xlwt.BIFFRecords.**DateModeRecord**(*from1904*)
Bases: xlwt.BIFFRecords.BiffRecord

This record specifies the base date for displaying date values. All dates are stored as count of days past this base date. In BIFF2-BIFF4 this record is part of the Calculation Settings Block. In BIFF5-BIFF8 it is stored in the Workbook Globals Substream.

Record DATEMODE, BIFF2-BIFF8:

Offset Size Contents 0 2 0 = Base is 1899-Dec-31 (the cell = 1 represents 1900-Jan-01)

1 = Base is 1904-Jan-01 (the cell = 1 represents 1904-Jan-02)

class xlwt.BIFFRecords.**DefColWidthRecord**(*def_width*)
Bases: xlwt.BIFFRecords.BiffRecord

This record specifies the default column width for columns that do not have a specific width set using the record COLINFO or COLWIDTH. This record has no effect, if a STANDARDWIDTH record is present in the file.

Record DEFCOLWIDTH, BIFF2-BIFF8:

Offset Size Contents 0 2 Column width in characters, using the width of the zero

character from default font (first FONT record in the file)

class xlwt.BIFFRecords.**DefaultRowHeightRecord**(*options*, *def_height*)
Bases: xlwt.BIFFRecords.BiffRecord

This record specifies the default height and default flags for rows that do not have a corresponding ROW record.

Record DEFAULTROWHEIGHT, BIFF3-BIFF8:

Offset Size Contents 0 2 Option flags:

Bit Mask Contents 0 0001H 1 = Row height and default font height do not match 1 0002H 1 = Row is hidden 2 0004H 1 = Additional space above the row 3 0008H 1 = Additional space below the row

2 2 Default height for unused rows, in twips = 1/20 of a point

class xlwt.BIFFRecords.**DeltaRecord**(*delta*)
Bases: xlwt.BIFFRecords.BiffRecord

This record is part of the Calculation Settings Block. It stores the maximum change of the result to exit an iteration.

Record DELTA, BIFF2-BIFF8:

Offset Size Contents 0 8 Maximum change in iteration

(IEEE 754 floating-point value, 64bit double precision)

class xlwt.BIFFRecords.**DimensionsRecord**(*first_used_row*, *last_used_row*, *first_used_col*, *last_used_col*)
Bases: xlwt.BIFFRecords.BiffRecord

Record DIMENSIONS, BIFF8:

Offset Size Contents 0 4 Index to first used row 4 4 Index to last used row, increased by 1 8 2 Index to first used column 10 2 Index to last used column, increased by 1 12 2 Not used

class xlwt.BIFFRecords.**EOFRecord**
Bases: xlwt.BIFFRecords.BiffRecord

class xlwt.BIFFRecords.**ExtSSTRecord**(*sst_stream_pos*, *str_placement*, *portions_len*)
Bases: xlwt.BIFFRecords.BiffRecord

This record occurs in conjunction with the SST record. It is used by Excel to create a hash table with stream offsets to the SST record to optimise string search operations. Excel may not shorten this record if strings are deleted from the shared string table, so the last part might contain invalid data. The stream indexes in this record divide the SST into portions containing a constant number of strings.

Record EXTSST, BIFF8:

Offset Size Contents 0 2 Number of strings in a portion, this number is >=8 2 var. List of OFFSET structures for all portions. Each OFFSET contains the following data:

Offset Size Contents 0 4 Absolute stream position of first string of the portion 4 2 Position of first string of the portion inside of current record,

including record header. This counter restarts at zero, if the SST record is continued with a CONTINUE record.

6 2 Not used

class xlwt.BIFFRecords.**ExternSheetRecord**(*refs*)
Bases: xlwt.BIFFRecords.BiffRecord

In BIFF8 the record stores a list with indexes to SUPBOOK records (list of REF structures, 6.100). See 5.10.3 for details about external references in BIFF8.

Record EXTERNSHEET, BIFF8: Offset Size Contents

0 2 Number of following REF structures (nm) 2 6nm List of nm REF structures. Each REF contains the following data:

Offset Size Contents 0 2 Index to SUPBOOK record 2 2 Index to first SUPBOOK sheet 4 2 Index to last SUPBOOK sheet

get()

class xlwt.BIFFRecords.**ExternnameRecord**(*options=0*, *index=0*, *name=None*, *fmla=None*)
Bases: xlwt.BIFFRecords.BiffRecord

Record EXTERNNAME for external names and Analysis add-in functions, BIFF5-BIFF8: Offset Size Contents 0 2 Option flags (see below) 2 2 0 for global names, or:

BIFF5: One-based index to EXTERNSHEET record containing the sheet name, BIFF8: One-based index to sheet list in preceding EXTERNALBOOK record.

4 2 Not used 6 var. BIFF5: Name (byte string, 8-bit string length, ?2.5.2).

BIFF8: Name (Unicode string, 8-bit string length, ?2.5.3). See DEFINEDNAME record (?5.33) for a list of built-in names, if the built-in flag is set in the option flags above.

var. var. Formula data (RPN token array, ?3)

Option flags for external names (BIFF5-BIFF8) Bit Mask Contents 0 0001H 0 = Standard name; 1 = Built-in name 1 0002H 0 = Manual link; 1 = Automatic link (DDE links and OLE links only) 2 0004H 1 = Picture link (DDE links and OLE links only) 3 0008H 1 = This is the "StdDocumentName" identifier (DDE links only) 4 0010H 1 = OLE link 14-5 7FE0H Clipboard format of last successful update (DDE links and OLE links only) 15 8000H 1 = Iconified picture link (BIFF8 OLE links only)

class xlwt.BIFFRecords.**FnGroupCountRecord**
Bases: xlwt.BIFFRecords.BiffRecord

class xlwt.BIFFRecords.**FontRecord**(*height*, *options*, *colour_index*, *weight*, *escapement*, *underline*, *family*, *charset*, *name*)
Bases: xlwt.BIFFRecords.BiffRecord

WARNING The font with index 4 is omitted in all BIFF versions. This means the first four fonts have zero-based indexes, and the fifth font and all following fonts are referenced with one-based indexes.

Offset Size Contents 0 2 Height of the font (in twips = 1/20 of a point) 2 2 Option flags:

Bit Mask Contents 0 0001H 1 = Characters are bold (redundant, see below) 1 0002H 1 = Characters are italic 2 0004H 1 = Characters are underlined (redundant, see below) 3 0008H 1 = Characters are struck out

0010H 1 = Outline 0020H 1 = Shadow

4 2 Colour index 6 2 Font weight (100-1000).

Standard values are 0190H (400) for normal text and 02BCH (700) for bold text.

8 2 Escapement type: 0000H = None 0001H = Superscript 0002H = Subscript

10 1 Underline type: 00H = None 01H = Single 21H = Single accounting 02H = Double 22H = Double accounting

11 1 Font family: 00H = None (unknown or don't care) 01H = Roman (variable width, serifed) 02H = Swiss (variable width, sans-serifed) 03H = Modern (fixed width, serifed or sans-serifed) 04H = Script (cursive) 05H = Decorative (specialised, i.e. Old English, Fraktur)

12 1 Character set: 00H = 0 = ANSI Latin 01H = 1 = System default 02H = 2 = Symbol 4DH = 77 = Apple Roman 80H = 128 = ANSI Japanese Shift-JIS 81H = 129 = ANSI Korean (Hangul) 82H = 130 = ANSI Korean (Johab) 86H = 134 = ANSI Chinese Simplified GBK 88H = 136 = ANSI Chinese Traditional BIG5 A1H = 161 = ANSI Greek A2H = 162 = ANSI Turkish A3H = 163 = ANSI Vietnamese B1H = 177 = ANSI Hebrew B2H = 178 = ANSI Arabic BAH = 186 = ANSI Baltic CCH = 204 = ANSI Cyrillic DEH = 222 = ANSI Thai EEH = 238 = ANSI Latin II (Central European) FFH = 255 = OEM Latin I

13 1 Not used 14 var. Font name:

BIFF5/BIFF7: Byte string, 8-bit string length BIFF8: Unicode string, 8-bit string length

The boldness and underline flags are still set in the options field, but not used on reading the font. Font weight and underline type are specified in separate fields instead.

class xlwt.BIFFRecords.**FooterRecord**(*footer_str*)
Bases: xlwt.BIFFRecords.BiffRecord

Semantic is equal to HEADER record

class xlwt.BIFFRecords.**FormulaRecord**(*row*, *col*, *xf_index*, *rpn*, *calc_flags=0*)
Bases: xlwt.BIFFRecords.BiffRecord

Offset Size Contents 0 2 Index to row 2 2 Index to column 4 2 Index to XF record 6 8 Result of the formula 14 2 Option flags:

Bit Mask Contents 0 0001H 1 = Recalculate always 1 0002H 1 = Calculate on open 3 0008H 1 = Part of a shared formula

16 4 Not used 20 var. Formula data (RPN token array)

class xlwt.BIFFRecords.**GridSetRecord**(*print_grid_changed*)
Bases: xlwt.BIFFRecords.BiffRecord

This record specifies if the option to print sheet grid lines (record PRINTGRIDLINES) has ever been changed.

Record GRIDSET, BIFF3-BIFF8:

Offset Size Contents 0 2 0 = Print grid lines option never changed

1 = Print grid lines option changed

class xlwt.BIFFRecords.**GutsRecord**(*row_gut_width*, *col_gut_height*, *row_visible_levels*, *col_visible_levels*)
Bases: xlwt.BIFFRecords.BiffRecord

This record contains information about the layout of outline symbols.

Record GUTS, BIFF3-BIFF8:

Offset Size Contents 0 2 Width of the area to display row outlines (left of the sheet), in pixel 2 2 Height of the area to display column outlines (above the sheet), in pixel 4 2 Number of visible row outline levels (used row levels + 1; or 0, if not used) 6 2 Number of visible column outline levels (used column levels + 1; or 0, if not used)

class xlwt.BIFFRecords.**HCenterRecord**(*is_horz_center*)
Bases: xlwt.BIFFRecords.BiffRecord

This record is part of the Page Settings Block. It specifies if the sheet is centred horizontally when printed.

Record HCENTER, BIFF3-BIFF8:

Offset Size Contents 0 2 0 = Print sheet left aligned

1 = Print sheet centred horizontally

class xlwt.BIFFRecords.**HeaderRecord**(*header_str*)
Bases: xlwt.BIFFRecords.BiffRecord

This record is part of the Page Settings Block. It specifies the page header string for the current worksheet. If this record is not present or completely empty (record size is 0), the sheet does not contain a page header.

Record HEADER for non-empty page header, BIFF2-BIFF8: Offset Size Contents 0 var. Page header string

BIFF2-BIFF7: Non-empty byte string, 8bit string length BIFF8: Non-empty Unicode string, 16bit string length

The header string may contain special commands, i.e. placeholders for the page number, current date, or text formatting attributes. These fields are represented by single letters (exception: font name and size, see below) with a leading ampersand ("&"). If the ampersand is part of the regular header text, it will be duplicated ("&&"). The page header is divided into 3 sections: the left, the centred, and the right section. Each section is introduced by a special command. All text and all commands following are part of the selected section. Each section starts with the text formatting specified in the default font (first FONT record in the file). Active formatting attributes from a previous section do not go into the next section.

The following table shows all available commands:

Command Contents && The "&" character itself &L Start of the left section &C Start of the centred section &R Start of the right section &P Current page number &N Page count &D Current date &T Current time &A Sheet name (BIFF5-BIFF8) &F File name without path &Z File path without file name (BIFF8X) &G Picture (BIFF8X) &B Bold on/off (BIFF2-BIFF4) &I Italic on/off (BIFF2-BIFF4) &U Underlining on/off &E Double underlining on/off (BIFF5-BIFF8) &S Strikeout on/off &X Superscript on/off (BIFF5-BIFF8) &Y Subscript on/off (BIFF5-BIFF8) &"<fontname>" Set new font <fontname> &"<fontname>,<fontstyle>"

Set new font with specified style <fontstyle>. The style <fontstyle> is in most cases one of "Regular", "Bold", "Italic", or "Bold Italic". But this setting is dependent on the used font, it may differ (localised style names, or "Standard", "Oblique", ...). (BIFF5-BIFF8)

&<fontheight> Set font height in points (<fontheight> is a decimal value). If this command is followed by a plain number to be printed in the header, it will be separated from the font height with a space character.

class xlwt.BIFFRecords.**HideObjRecord**
Bases: xlwt.BIFFRecords.BiffRecord

This record specifies whether and how to show objects in the workbook.

Record HIDEOBJ, BIFF3-BIFF8: Offset Size Contents 0 2 Viewing mode for objects:

0 = Show all objects 1 = Show placeholders 2 = Do not show objects

class xlwt.BIFFRecords.**HorizontalPageBreaksRecord**(*breaks_list*)
Bases: xlwt.BIFFRecords.BiffRecord

This record is part of the Page Settings Block. It contains all horizontal manual page breaks.

Record HORIZONTALPAGEBREAKS, BIFF8: Offset Size Contents 0 2 Number of following row index structures (nm) 2 6nm List of nm row index structures. Each row index

structure contains: Offset Size Contents 0 2 Index to first row below the page break 2 2 Index to first column of this page break 4 2 Index to last column of this page break

The row indexes in the lists must be ordered ascending. If in BIFF8 a row contains several page breaks, they must be ordered ascending by start column index.

class xlwt.BIFFRecords.**InteraceEndRecord**
Bases: xlwt.BIFFRecords.BiffRecord

class xlwt.BIFFRecords.**InteraceHdrRecord**
Bases: xlwt.BIFFRecords.BiffRecord

class xlwt.BIFFRecords.**InternalReferenceSupBookRecord**(*num_sheets*)
Bases: xlwt.BIFFRecords.SupBookRecord

In each file occurs a SUPBOOK that is used for internal 3D references. It stores the number of sheets of the own document.

Record SUPBOOK for 3D references, BIFF8: Offset Size Contents

0 2 Number of sheets in this document 2 2 01H 04H (relict of BIFF5/BIFF7, the byte string "<04H>", see 3.9.1)

class xlwt.BIFFRecords.**IterationRecord**(*iterations_on*)
Bases: xlwt.BIFFRecords.BiffRecord

This record is part of the Calculation Settings Block. It stores if iterations are allowed while calculating recursive formulas.

Record ITERATION, BIFF2-BIFF8:

Offset Size Contents 0 2 0 = Iterations off; 1 = Iterations on

class xlwt.BIFFRecords.**LabelSSTRecord**(*row*, *col*, *xf_idx*, *sst_idx*)
Bases: xlwt.BIFFRecords.BiffRecord

This record represents a cell that contains a string. It replaces the LABEL record and RSTRING record used in BIFF2-BIFF7.

class xlwt.BIFFRecords.**LeftMarginRecord**(*margin*)
Bases: xlwt.BIFFRecords.BiffRecord

This record is part of the Page Settings Block. It contains the left page margin of the current worksheet.

Record LEFTMARGIN, BIFF2-BIFF8:

Offset Size Contents 0 8 Left page margin in inches

(IEEE 754 floating-point value, 64bit double precision)

class xlwt.BIFFRecords.**MMSRecord**
Bases: xlwt.BIFFRecords.BiffRecord

class xlwt.BIFFRecords.**MergedCellsRecord**(*merged_list*)
Bases: xlwt.BIFFRecords.BiffRecord

This record contains all merged cell ranges of the current sheet.

Record MERGEDCELLS, BIFF8:

Offset Size Contents 0 var. Cell range address list with all merged ranges

A cell range address list consists of a field with the number of ranges and the list of the range addresses.

Cell range address list, BIFF8:

Offset Size Contents 0 2 Number of following cell range addresses (nm) 2 8*nm List of nm cell range addresses

Cell range address, BIFF8:

Offset Size Contents 0 2 Index to first row 2 2 Index to last row 4 2 Index to first column 6 2 Index to last column

get()

class xlwt.BIFFRecords.**MulBlankRecord**(*row*, *first_col*, *last_col*, *xf_index*)
Bases: xlwt.BIFFRecords.BiffRecord

This record represents a cell range of empty cells. All cells are located in the same row.

Record MULBLANK, BIFF5-BIFF8:

Offset Size Contents 0 2 Index to row 2 2 Index to first column (fc) 4 2*nc List of nc=lc-fc+1 16-bit indexes to XF records 4+2*nc 2 Index to last column (lc)

class xlwt.BIFFRecords.**NameRecord**(*options*, *keyboard_shortcut*, *name*, *sheet_index*, *rpn*, *menu_text=''*, *desc_text=''*, *help_text=''*, *status_text=''*)
Bases: xlwt.BIFFRecords.BiffRecord

This record is part of a Link Table. It contains the name and the token array of an internal defined name. Token arrays of defined names contain tokens with aberrant token classes.

Record NAME, BIFF5/BIFF7: Offset Size Contents

0 2 Option flags, see below 2 1 Keyboard shortcut (only for command macro names, see below) 3 1 Length of the name (character count, ln) 4 2 Size of the formula data (sz) 6 2 0 = Global name, otherwise index to EXTERNSHEET record (one-based) 8 2 0 = Global name, otherwise index to sheet (one-based)

10 1 Length of menu text (character count, lm) 11 1 Length of description text (character count, ld) 12 1 Length of help topic text (character count, lh) 13 1 Length of status bar text (character count, ls) 14 ln Character array of the name

14+ln sz Formula data (RPN token array without size field, 4)

14+ln+sz lm Character array of menu text

var. ld Character array of description text var. lh Character array of help topic text var. ls Character array of status bar text

Record NAME, BIFF8: Offset Size Contents

0 2 Option flags, see below 2 1 Keyboard shortcut (only for command macro names, see below) 3 1 Length of the name (character count, ln) 4 2 Size of the formula data (sz) 6 2 Not used 8 2 0 = Global name, otherwise index to sheet (one-based)

10 1 Length of menu text (character count, lm) 11 1 Length of description text (character count, ld) 12 1 Length of help topic text (character count, lh) 13 1 Length of status bar text (character count, ls) 14 var. Name (Unicode string without length field, 3.4)

var. sz Formula data (RPN token array without size field, 4)

[var.] var. (optional, only if lm > 0) Menu text (Unicode string without length field, 3.4) [var.] var. (optional, only if ld > 0) Description text (Unicode string without length field, 3.4) [var.] var. (optional, only if lh > 0) Help topic text (Unicode string without length field, 3.4) [var.] var. (optional, only if ls > 0) Status bar text (Unicode string without length field, 3.4)

class xlwt.BIFFRecords.**NumberFormatRecord**(*idx*, *fmtstr*)

Bases: xlwt.BIFFRecords.BiffRecord

Record FORMAT, BIFF8: Offset Size Contents 0 2 Format index used in other records 2 var. Number format string (Unicode string, 16-bit string length)

From BIFF5 on, the built-in number formats will be omitted. The built-in formats are dependent on the current regional settings of the operating system. The following table shows which number formats are used by default in a US-English environment. All indexes from 0 to 163 are reserved for built-in formats. The first user-defined format starts at 164.

The built-in number formats, BIFF5-BIFF8

Index Type Format string 0 General General 1 Decimal 0 2 Decimal 0.00 3 Decimal #,##0 4 Decimal #,##0.00 5 Currency "$"#,##0_);("$"#,## 6 Currency "$"#,##0_);[Red]("$"#,## 7 Currency "$"#,##0.00_);("$"#,## 8 Currency "$"#,##0.00_);[Red]("$"#,## 9 Percent 0% 10 Percent 0.00% 11 Scientific 0.00E+00 12 Fraction # ?/? 13 Fraction # ??/?? 14 Date M/D/YY 15 Date D-MMM-YY 16 Date D-MMM 17 Date MMM-YY 18 Time h:mm AM/PM 19 Time h:mm:ss AM/PM 20 Time h:mm 21 Time h:mm:ss 22 Date/Time M/D/YY h:mm 37 Account _(#,##0_);(#,##0) 38 Account _(#,##0_);[Red](#,##0) 39 Account _(#,##0.00_);(#,##0.00) 40 Account _(#,##0.00_);[Red](#,##0.00) 41 Currency _("$"* #,##0_);_("$"* (#,##0);_("$"* "-"_);_(@_) 42 Currency _(* #,##0_);_(* (#,##0);_(* "-"_);_(@_) 43 Currency _("$"* #,##0.00_);_("$"* (#,##0.00);_("$"* "-"??_);_(@_) 44 Currency _(* #,##0.00_);_(* (#,##0.00);_(* "-"??_);_(@_) 45 Time mm:ss 46 Time [h]:mm:ss 47 Time mm:ss.0 48 Scientific ##0.0E+0 49 Text @

class xlwt.BIFFRecords.**NumberRecord**(*row*, *col*, *xf_index*, *number*)
Bases: xlwt.BIFFRecords.BiffRecord

This record represents a cell that contains an IEEE-754 floating-point value.

class xlwt.BIFFRecords.**ObjectProtectRecord**(*objprotect*)
Bases: xlwt.BIFFRecords.BiffRecord

This record is part of the worksheet/workbook protection. It determines whether the objects of the current sheet are protected. Object protection is not active, if this record is omitted.

class xlwt.BIFFRecords.**PaletteRecord**
Bases: xlwt.BIFFRecords.BiffRecord

This record contains the definition of all user-defined colours available for cell and object formatting.

Record PALETTE, BIFF3-BIFF8:

Offset Size Contents 0 2 Number of following colours (nm). Contains 16 in BIFF3-BIFF4 and 56 in BIFF5-BIFF8. 2 4*nm List of nm RGB colours

The following table shows how colour indexes are used in other records:

Colour index Resulting colour or internal list index 00H Built-in Black (R = 00H, G = 00H, B = 00H) 01H Built-in White (R = FFH, G = FFH, B = FFH) 02H Built-in Red (R = FFH, G = 00H, B = 00H) 03H Built-in Green (R = 00H, G = FFH, B = 00H) 04H Built-in Blue (R = 00H, G = 00H, B = FFH) 05H Built-in Yellow (R = FFH, G = FFH, B = 00H) 06H Built-in Magenta (R = FFH, G = 00H, B = FFH) 07H Built-in Cyan (R = 00H, G = FFH, B = FFH) 08H First user-defined colour from the PALETTE record (entry 0 from record colour list) .....

17H (BIFF3-BIFF4) Last user-defined colour from the PALETTE record (entry 15 or 55 from record colour list) 3FH (BIFF5-BIFF8)

18H (BIFF3-BIFF4) System window text colour for border lines (used in records XF, CF, and 40H (BIFF5-BIFF8) WINDOW2 (BIFF8 only))

19H (BIFF3-BIFF4) System window background colour for pattern background (used in records XF, and CF) 41H (BIFF5-BIFF8)

43H System face colour (dialogue background colour) 4DH System window text colour for chart border lines 4EH System window background colour for chart areas 4FH Automatic colour for chart border lines (seems to be always Black) 50H System ToolTip background colour (used in note objects) 51H System ToolTip text colour (used in note objects) 7FFFH System window text colour for fonts (used in records FONT, EFONT, and CF)

class xlwt.BIFFRecords.**PanesRecord**(*px*, *py*, *first_row_bottom*, *first_col_right*, *active_pane*)
Bases: xlwt.BIFFRecords.BiffRecord

This record stores the position of window panes. It is part of the Sheet View Settings Block. If the sheet does not contain any splits, this record will not occur. A sheet can be split in two different ways, with unfrozen panes or with frozen panes. A flag in the WINDOW2 record specifies, if the panes are frozen, which affects the contents of this record.

Record PANE, BIFF2-BIFF8: Offset Size Contents 0 2 Position of the vertical split

(px, 0 = No vertical split): Unfrozen pane: Width of the left pane(s) (in twips = 1/20 of a point) Frozen pane: Number of visible columns in left pane(s)

2 2 Position of the horizontal split (py, 0 = No horizontal split): Unfrozen pane: Height of the top pane(s) (in twips = 1/20 of a point) Frozen pane: Number of visible rows in top pane(s)

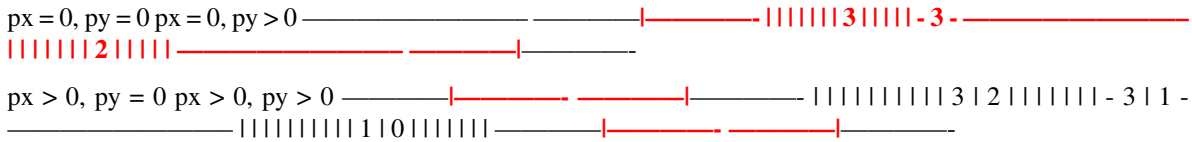
4 2 Index to first visible row in bottom pane(s)

6 2 Index to first visible column in right pane(s)

8 1 Identifier of pane with active cell cursor

[9] 1 Not used (BIFF5-BIFF8 only, not written in BIFF2-BIFF4)

If the panes are frozen, pane 0 is always active, regardless of the cursor position. The correct identifiers for all possible combinations of visible panes are shown in the following pictures.



class xlwt.BIFFRecords.**PasswordRecord**(*passwd=''*)
Bases: xlwt.BIFFRecords.BiffRecord

This record is part of the worksheet/workbook protection. It stores a 16-bit hash value, calculated from the worksheet or workbook protection password.

passwd_hash(*plaintext*)
Based on the algorithm provided by Daniel Rentz of OpenOffice.

class xlwt.BIFFRecords.**PrecisionRecord**(*use_real_values*)
Bases: xlwt.BIFFRecords.BiffRecord

This record stores if formulas use the real cell values for calculation or the values displayed on the screen. In BIFF2- BIFF4 this record is part of the Calculation Settings Block. In BIFF5-BIFF8 it is stored in the Workbook Globals Substream.

Record PRECISION, BIFF2-BIFF8:

Offset Size Contents 0 2 0 = Use displayed values;

1 = Use real cell values

class xlwt.BIFFRecords.**PrintGridLinesRecord**(*print_grid*)
Bases: xlwt.BIFFRecords.BiffRecord

This record stores if sheet grid lines will be printed.

Record PRINTGRIDLINES, BIFF2-BIFF8:

Offset Size Contents 0 2 0 = Do not print sheet grid lines;

1 = Print sheet grid lines

class xlwt.BIFFRecords.**PrintHeadersRecord**(*print_headers*)
Bases: xlwt.BIFFRecords.BiffRecord

This record stores if the row and column headers (the areas with row numbers and column letters) will be printed.

Record PRINTHEADERS, BIFF2-BIFF8:

Offset Size Contents 0 2 0 = Do not print row/column headers;

1 = Print row/column headers

class xlwt.BIFFRecords.**Prot4RevPassRecord**
Bases: xlwt.BIFFRecords.BiffRecord

class xlwt.BIFFRecords.**Prot4RevRecord**
Bases: xlwt.BIFFRecords.BiffRecord

class xlwt.BIFFRecords.**ProtectRecord**(*protect*)
Bases: xlwt.BIFFRecords.BiffRecord

This record is part of the worksheet/workbook protection. It specifies whether a worksheet or a workbook is protected against modification. Protection is not active, if this record is omitted.

class xlwt.BIFFRecords.**RKRecord**(*row*, *col*, *xf_index*, *rk_encoded*)
Bases: xlwt.BIFFRecords.BiffRecord

This record represents a cell that contains an RK value (encoded integer or floating-point value). If a floating-point value cannot be encoded to an RK value, a NUMBER record will be written.

class xlwt.BIFFRecords.**RefModeRecord**(*ref_mode*)
Bases: xlwt.BIFFRecords.BiffRecord

This record is part of the Calculation Settings Block. It stores which method is used to show cell addresses in formulas. The "RC" mode uses numeric indexes for rows and columns, i.e. "R(1)C(-1)", or "R1C1:R2C2". The "A1" mode uses characters for columns and numbers for rows, i.e. "B1", or "$A$1:$B$2".

Record REFMODE, BIFF2-BIFF8:

Offset Size Contents 0 2 0 = RC mode; 1 = A1 mode

class xlwt.BIFFRecords.**RefreshAllRecord**
Bases: xlwt.BIFFRecords.BiffRecord

class xlwt.BIFFRecords.**RightMarginRecord**(*margin*)
Bases: xlwt.BIFFRecords.BiffRecord

This record is part of the Page Settings Block. It contains the right page margin of the current worksheet.

Offset Size Contents 0 8 Right page margin in inches

(IEEE 754 floating-point value, 64?bit double precision)

class xlwt.BIFFRecords.**RowRecord**(*index*, *first_col*, *last_col*, *height_options*, *options*)
Bases: xlwt.BIFFRecords.BiffRecord

This record contains the properties of a single row in a sheet. Rows and cells in a sheet are divided into blocks of 32 rows.

Record ROW, BIFF3-BIFF8:

Offset Size Contents 0 2 Index of this row 2 2 Index to column of the first cell which is described by a cell record 4 2 Index to column of the last cell which is described by a cell record,

increased by 1

6 2 Bit Mask Contents 14-0 7FFFH Height of the row, in twips = 1/20 of a point 15 8000H 0 = Row has custom height; 1 = Row has default height

8 2 Not used 10 2 In BIFF3-BIFF4 this field contains a relative offset

to calculate stream position of the first cell record for this row. In BIFF5-BIFF8 this field is not used anymore, but the DBCELL record instead.

12 4 Option flags and default row formatting: Bit Mask Contents 2-0 00000007H Outline level of the row 4 00000010H 1 = Outline group starts or ends here (depending

on where the outline buttons are located, see WSBOOL record), and is collapsed

5 00000020H 1 = Row is hidden (manually, or by a filter or outline group) 6 00000040H 1 = Row height and default font height do not match 7 00000080H 1 = Row has explicit default format (fl) 8 00000100H

Always 1 27-16 0FFF0000H If fl=1: Index to default XF record 28 10000000H 1 = Additional space above the row. This flag is set,

if the upper border of at least one cell in this row or if the lower border of at least one cell in the row above is formatted with a thick line style. Thin and medium line styles are not taken into account.

29 20000000H 1 = Additional space below the row. This flag is set, if the lower border of at least one cell in this row or if the upper border of at least one cell in the row below is formatted with a medium or thick line style. Thin line styles are not taken into account.

class xlwt.BIFFRecords.**SSTRecord**
Bases: xlwt.BIFFRecords.BiffRecord

This record contains a list of all strings used anywhere in the workbook. Each string occurs only once. The workbook uses indexes into the list to reference the strings.

Record SST, BIFF8: Offset Size Contents 0 4 Total number of strings in the workbook (see below) 4 4 Number of following strings (nm) 8 var. List of nm Unicode strings, 16-bit string length

The first field of the SST record counts the total occurrence of strings in the workbook. For instance, the string AAA is used 3 times and the string BBB is used 2 times. The first field contains 5 and the second field contains 2, followed by the two strings.

class xlwt.BIFFRecords.**SaveRecalcRecord**(*recalc*)
Bases: xlwt.BIFFRecords.BiffRecord

This record is part of the Calculation Settings Block. It contains the "Recalculate before save" option in Excel's calculation settings dialogue.

Record SAVERECALC, BIFF3-BIFF8:

Offset Size Contents 0 2 0 = Do not recalculate;

1 = Recalculate before saving the document

class xlwt.BIFFRecords.**ScenProtectRecord**(*scenprotect*)
Bases: xlwt.BIFFRecords.BiffRecord

This record is part of the worksheet/workbook protection. It determines whether the scenarios of the current sheet are protected. Scenario protection is not active, if this record is omitted.

class xlwt.BIFFRecords.**SetupPageRecord**(*paper*, *scaling*, *start_num*, *fit_width_to*, *fit_height_to*, *options*, *hres*, *vres*, *header_margin*, *footer_margin*, *num_copies*)
Bases: xlwt.BIFFRecords.BiffRecord

This record is part of the Page Settings Block. It stores the page format settings of the current sheet. The pages may be scaled in percent or by using an absolute number of pages. This setting is located in the WSBOOL record. If pages are scaled in percent, the scaling factor in this record is used, otherwise the "Fit to pages" values. One of the "Fit to pages" values may be 0. In this case the sheet is scaled to fit only to the other value.

Record SETUP, BIFF5-BIFF8:

Offset Size Contents 0 2 Paper size (see below) 2 2 Scaling factor in percent 4 2 Start page number 6 2 Fit worksheet width to this number of pages

(0 = use as many as needed)

8 2 Fit worksheet height to this number of pages (0 = use as many as needed)

10 2 Option flags: Bit Mask Contents 0 0001H 0 = Print pages in columns

1 = Print pages in rows

1 0002H 0 = Landscape 1 = Portrait

2 0004H 1 = Paper size, scaling factor, paper orientation (portrait/landscape), print resolution and number of copies are not initialised

3 0008H 0 = Print coloured 1 = Print black and white

4 0010H 0 = Default print quality 1 = Draft quality

5 0020H 0 = Do not print cell notes 1 = Print cell notes

6 0040H 0 = Paper orientation setting is valid 1 = Paper orientation setting not initialised

7 0080H 0 = Automatic page numbers 1 = Use start page number

The following flags are valid for BIFF8 only: 9 0200H 0 = Print notes as displayed

1 = Print notes at end of sheet

11-10 0C00H 00 = Print errors as displayed 01 = Do not print errors 10 = Print errors as "–" 11 = Print errors as "#N/A!"

12 2 Print resolution in dpi 14 2 Vertical print resolution in dpi 16 8 Header margin (IEEE 754 floating-point value,

64bit double precision)

24 8 Footer margin (IEEE 754 floating-point value, 64bit double precision)

32 2 Number of copies to print

PAPER TYPES:

Index Paper type Paper size 0 Undefined 1 Letter 8 1/2" x 11" 2 Letter small 8 1/2" x 11" 3 Tabloid 11" x 17" 4 Ledger 17" x 11" 5 Legal 8 1/2" x 14" 6 Statement 5 1/2" x 8 1/2" 7 Executive 7 1/4" x 10 1/2" 8 A3 297mm x 420mm 9 A4 210mm x 297mm 10 A4 small 210mm x 297mm 11 A5 148mm x 210mm 12 B4 (JIS) 257mm x 364mm 13 B5 (JIS) 182mm x 257mm 14 Folio 8 1/2" x 13" 15 Quarto 215mm x 275mm 16 10x14 10" x 14" 17 11x17 11" x 17" 18 Note 8 1/2" x 11" 19 Envelope #9 3 7/8" x 8 7/8" 20 Envelope #10 4 1/8" x 9 1/2" 21 Envelope #11 4 1/2" x 10 3/8" 22 Envelope #12 4 3/4" x 11" 23 Envelope #14 5" x 11 1/2" 24 C 17" x 22" 25 D 22" x 34" 26 E 34" x 44" 27 Envelope DL 110mm x 220mm 28 Envelope C5 162mm x 229mm 29 Envelope C3 324mm x 458mm 30 Envelope C4 229mm x 324mm 31 Envelope C6 114mm x 162mm 32 Envelope C6/C5 114mm x 229mm 33 B4 (ISO) 250mm x 353mm 34 B5 (ISO) 176mm x 250mm 35 B6 (ISO) 125mm x 176mm 36 Envelope Italy 110mm x 230mm 37 Envelope Monarch 3 7/8" x 7 1/2" 38 63/4 Envelope 3 5/8" x 6 1/2" 39 US Standard Fanfold 14 7/8" x 11" 40 German Std. Fanfold 8 1/2" x 12" 41 German Legal Fanfold 8 1/2" x 13" 42 B4 (ISO) 250mm x 353mm 43 Japanese Postcard 100mm x 148mm 44 9x11 9" x 11" 45 10x11 10" x 11" 46 15x11 15" x 11" 47 Envelope Invite 220mm x 220mm 48 Undefined 49 Undefined 50 Letter Extra 9 1/2" x 12" 51 Legal Extra 9 1/2" x 15" 52 Tabloid Extra 11 11/16" x 18" 53 A4 Extra 235mm x 322mm 54 Letter Transverse 8 1/2" x 11" 55 A4 Transverse 210mm x 297mm 56 Letter Extra Transv. 9 1/2" x 12" 57 Super A/A4 227mm x 356mm 58 Super B/A3 305mm x 487mm 59 Letter Plus 8 1/2" x 12 11/16" 60 A4 Plus 210mm x 330mm 61 A5 Transverse 148mm x 210mm 62 B5 (JIS) Transverse 182mm x 257mm 63 A3 Extra 322mm x 445mm 64 A5 Extra 174mm x 235mm 65 B5 (ISO) Extra 201mm x 276mm 66 A2 420mm x 594mm 67 A3 Transverse 297mm x 420mm 68 A3 Extra Transverse 322mm x 445mm 69 Dbl. Japanese Postcard 200mm x 148mm 70 A6 105mm x 148mm 71 72 73 74 75 Letter Rotated 11" x 8 1/2" 76 A3 Rotated 420mm x 297mm 77 A4 Rotated 297mm x 210mm 78 A5 Rotated 210mm x 148mm 79 B4 (JIS) Rotated 364mm x 257mm 80 B5 (JIS) Rotated 257mm x 182mm 81 Japanese Postcard Rot. 148mm x 100mm 82 Dbl. Jap. Postcard Rot. 148mm x 200mm 83 A6 Rotated 148mm x 105mm 84 85 86 87 88 B6 (JIS) 128mm x 182mm 89 B6 (JIS) Rotated 182mm x 128mm 90 12x11 12" x 11"

class xlwt.BIFFRecords.**SharedStringTable**(*encoding*)
Bases: object

add_str(*s*)

del_str(*idx*)

get_biff_record()

str_index(*s*)

class xlwt.BIFFRecords.**StyleRecord**
Bases: xlwt.BIFFRecords.BiffRecord

STYLE record for user-defined cell styles, BIFF3-BIFF8: Offset Size Contents 0 2 Bit Mask Contents

11-0 0FFFH Index to style XF record 15 8000H Always 0 for user-defined styles

2 var. BIFF2-BIFF7: Non-empty byte string, 8-bit string length BIFF8: Non-empty Unicode string, 16-bit string length

STYLE record for built-in cell styles, BIFF3-BIFF8: Offset Size Contents 0 2 Bit Mask Contents

11-0 0FFFH Index to style XF record 15 8000H Always 1 for built-in styles

2 1 Identifier of the built-in cell style: 00H = Normal 01H = RowLevel_lv (see next field) 02H = ColLevel_lv (see next field) 03H = Comma 04H = Currency 05H = Percent 06H = Comma [0] (BIFF4-BIFF8) 07H = Currency [0] (BIFF4-BIFF8) 08H = Hyperlink (BIFF8) 09H = Followed Hyperlink (BIFF8)

3 1 Level for RowLevel or ColLevel style (zero-based, lv), FFH otherwise

The RowLevel and ColLevel styles specify the formatting of subtotal cells in a specific outline level. The level is specified by the last field in the STYLE record. Valid values are 0-6 for the outline levels 1-7.

class xlwt.BIFFRecords.**SupBookRecord**
Bases: xlwt.BIFFRecords.BiffRecord

This record mainly stores the URL of an external document and a list of sheet names inside this document. Furthermore it is used to store DDE and OLE object links, or to indicate an internal 3D reference or an add-in function. See 5.10.3 for details about external references in BIFF8.

class xlwt.BIFFRecords.**TabIDRecord**(*sheetcount*)
Bases: xlwt.BIFFRecords.BiffRecord

class xlwt.BIFFRecords.**TopMarginRecord**(*margin*)
Bases: xlwt.BIFFRecords.BiffRecord

This record is part of the Page Settings Block. It contains the top page margin of the current worksheet.

Offset Size Contents 0 8 Top page margin in inches

(IEEE 754 floating-point value, 64?bit double precision)

class xlwt.BIFFRecords.**UseSelfsRecord**
Bases: xlwt.BIFFRecords.BiffRecord

This record specifies if the formulas in the workbook can use natural language formulas. This type of formula can refer to cells by its content or the content of the column or row header cell.

Record USESELFS, BIFF8:

Offset Size Contents 0 2 0 = Do not use natural language formulas

1 = Use natural language formulas

class xlwt.BIFFRecords.**VCenterRecord**(*is_vert_center*)
Bases: xlwt.BIFFRecords.BiffRecord

This record is part of the Page Settings Block. It specifies if the sheet is centred vertically when printed.

Record VCENTER, BIFF3-BIFF8:

Offset Size Contents 0 2 0 = Print sheet aligned at top page border

1 = Print sheet vertically centred

class xlwt.BIFFRecords.**VerticalPageBreaksRecord**(*breaks_list*)
Bases: xlwt.BIFFRecords.BiffRecord

This record is part of the Page Settings Block. It contains all vertical manual page breaks.

Record VERTICALPAGEBREAKS, BIFF8: Offset Size Contents 0 2 Number of following column index structures (nm) 2 6nm List of nm column index structures. Each column index

structure contains: Offset Size Contents 0 2 Index to first column following the page

break

2 2 Index to first row of this page break 4 2 Index to last row of this page break

The column indexes in the lists must be ordered ascending. If in BIFF8 a column contains several page breaks, they must be ordered ascending by start row index.

class xlwt.BIFFRecords.**WSBoolRecord**(*options*)
Bases: xlwt.BIFFRecords.BiffRecord

This record stores a 16 bit value with Boolean options for the current sheet. From BIFF5 on the "Save external linked values" option is moved to the record BOOKBOOL.

Option flags of record WSBOOL, BIFF3-BIFF8:

Bit Mask Contents 0 0001H 0 = Do not show automatic page breaks

1 = Show automatic page breaks

4 0010H 0 = Standard sheet 1 = Dialogue sheet (BIFF5-BIFF8)

5 0020H 0 = No automatic styles in outlines 1 = Apply automatic styles to outlines

6 0040H 0 = Outline buttons above outline group 1 = Outline buttons below outline group

7 0080H 0 = Outline buttons left of outline group 1 = Outline buttons right of outline group

8 0100H 0 = Scale printout in percent 1 = Fit printout to number of pages

9 0200H 0 = Save external linked values (BIFF3?BIFF4 only) 1 = Do not save external linked values (BIFF3?BIFF4 only)

10 0400H 0 = Do not show row outline symbols 1 = Show row outline symbols

11 0800H 0 = Do not show column outline symbols 1 = Show column outline symbols

13-12 3000H These flags specify the arrangement of windows. They are stored in BIFF4 only. 00 = Arrange windows tiled 01 = Arrange windows horizontal 10 = Arrange windows vertical112 = Arrange windows cascaded

The following flags are valid for BIFF4-BIFF8 only: 14 4000H 0 = Standard expression evaluation

1 = Alternative expression evaluation

15 8000H 0 = Standard formula entries 1 = Alternative formula entries

class xlwt.BIFFRecords.**Window1Record**(*hpos_twips*, *vpos_twips*, *width_twips*, *height_twips*, *flags*, *active_sheet*, *first_tab_index*, *selected_tabs*, *tab_width*)

Bases: xlwt.BIFFRecords.BiffRecord

Offset Size Contents 0 2 Horizontal position of the document window (in twips = 1/20 of a point) 2 2 Vertical position of the document window (in twips = 1/20 of a point) 4 2 Width of the document window (in twips = 1/20 of a point) 6 2 Height of the document window (in twips = 1/20 of a point) 8 2 Option flags:

Bits Mask Contents 0 0001H 0 = Window is visible 1 = Window is hidden 1 0002H 0 = Window is open 1 = Window is minimised 3 0008H 0 = Horizontal scroll bar hidden 1 = Horizontal scroll bar visible 4 0010H 0 = Vertical scroll bar hidden 1 = Vertical scroll bar visible 5 0020H 0 = Worksheet tab bar hidden 1 = Worksheet tab bar visible

10 2 Index to active (displayed) worksheet 12 2 Index of first visible tab in the worksheet tab bar 14 2 Number of selected worksheets (highlighted in the worksheet tab bar) 16 2 Width of worksheet tab bar (in 1/1000 of window width). The remaining space is used by the

horizontal scrollbar.

class xlwt.BIFFRecords.**Window2Record**(*options*, *first_visible_row*, *first_visible_col*, *grid_colour*, *preview_magn*, *normal_magn*, *scl_magn*)

Bases: xlwt.BIFFRecords.BiffRecord

Record WINDOW2, BIFF8:

Offset Size Contents 0 2 Option flags (see below) 2 2 Index to first visible row 4 2 Index to first visible column 6 2 Colour index of grid line colour. Note that in BIFF2-BIFF7 an RGB colour is

written instead.

8 2 Not used 10 2 Cached magnification factor in page break preview (in percent); 0 = Default (60%) 12 2 Cached magnification factor in normal view (in percent); 0 = Default (100%) 14 4 Not used

In BIFF8 this record stores used magnification factors for page break preview and normal view. These values are used to restore the magnification, when the view is changed. The real magnification of the currently active view is stored in the SCL record. The type of the active view is stored in the option flags field (see below).

0 0001H 0 = Show formula results 1 = Show formulas 1 0002H 0 = Do not show grid lines 1 = Show grid lines 2 0004H 0 = Do not show sheet headers 1 = Show sheet headers 3 0008H 0 = Panes are not frozen 1 = Panes are frozen (freeze) 4 0010H 0 = Show zero values as empty cells 1 = Show zero values 5 0020H 0 = Manual grid line colour 1 = Automatic grid line colour 6 0040H 0 = Columns from left to right 1 = Columns from right to left 7 0080H 0 = Do not show outline symbols 1 = Show outline symbols 8 0100H 0 = Keep splits if pane freeze is removed 1 = Remove splits if pane freeze is removed 9 0200H 0 = Sheet not selected 1 = Sheet selected (BIFF5-BIFF8)

10 0400H 0 = Sheet not visible 1 = Sheet visible (BIFF5-BIFF8) 11 0800H 0 = Show in normal view 1 = Show in page break preview (BIFF8)

The freeze flag specifies, if a following PANE record describes unfrozen or frozen panes.

*** This class appends the optional SCL record ***

Record SCL, BIFF4-BIFF8:

This record stores the magnification of the active view of the current worksheet. In BIFF8 this can be either the normal view or the page break preview. This is determined in the WINDOW2 record. The SCL record is part of the Sheet View Settings Block.

Offset Size Contents 0 2 Numerator of the view magnification fraction (num) 2 2 Denumerator [denominator] of the view magnification fraction (den) The magnification is stored as reduced fraction. The magnification results from num/den.

SJM note: Excel expresses (e.g.) 25% in reduced form i.e. 1/4. Reason unknown. This code writes 25/100, and Excel is happy with that.

get()

class xlwt.BIFFRecords.**WindowProtectRecord**(*wndprotect*)
Bases: xlwt.BIFFRecords.BiffRecord

This record is part of the worksheet/workbook protection. It determines whether the window configuration of this document is protected. Window protection is not active, if this record is omitted.

class xlwt.BIFFRecords.**WriteAccessRecord**(*owner*)
Bases: xlwt.BIFFRecords.BiffRecord

This record is part of the file protection. It contains the name of the user that has saved the file. The user name is always stored as an equal-sized string. All unused characters after the name are filled with space characters. It is not required to write the mentioned string length. Every other length will be accepted too.

class xlwt.BIFFRecords.**XFRecord**(*xf*, *xftype='cell'*)
Bases: xlwt.BIFFRecords.BiffRecord

XF_TYPE_PROT XF Type and Cell Protection (3 Bits), BIFF3-BIFF8 These 3 bits are part of a specific data byte. Bit Mask Contents 0 01H 1 = Cell is locked 1 02H 1 = Formula is hidden 2 04H 0 = Cell XF; 1 = Style XF

XF_USED_ATTRIB Attributes Used from Parent Style XF (6 Bits), BIFF3-BIFF8 Each bit describes the validity of a specific group of attributes. In cell XFs a cleared bit means the attributes of the parent style XF are used (but only if the attributes are valid there), a set bit means the attributes of this XF are used. In style XFs a cleared bit means the attribute setting is valid, a set bit means the attribute should be ignored. Bit Mask Contents 0 01H Flag for number format 1 02H Flag for font 2 04H Flag for horizontal and vertical alignment, text wrap, indentation, orientation, rotation, and

text direction

3 08H Flag for border lines 4 10H Flag for background area style 5 20H Flag for cell protection (cell locked and formula hidden)

XF_HOR_ALIGN Horizontal Alignment (3 Bits), BIFF2-BIFF8 The horizontal alignment consists of 3 bits and is part of a specific data byte. Value Horizontal alignment 00H General 01H Left 02H Centred 03H Right 04H Filled 05H Justified (BIFF4-BIFF8X) 06H Centred across selection (BIFF4-BIFF8X) 07H Distributed (BIFF8X)

XF_VERT_ALIGN Vertical Alignment (2 or 3 Bits), BIFF4-BIFF8 The vertical alignment consists of 2 bits (BIFF4) or 3 bits (BIFF5-BIFF8) and is part of a specific data byte. Vertical alignment is not available in BIFF2 and BIFF3. Value Vertical alignment 00H Top 01H Centred 02H Bottom 03H Justified (BIFF5-BIFF8X) 04H Distributed (BIFF8X)

XF_ORIENTATION Text Orientation (2 Bits), BIFF4-BIFF7 In the BIFF versions BIFF4-BIFF7, text can be rotated in steps of 90 degrees or stacked. The orientation mode consists of 2 bits and is part of a specific data byte. In BIFF8 a rotation angle occurs instead of these flags. Value Text orientation 00H Not rotated 01H Letters are stacked top-to-bottom, but not rotated 02H Text is rotated 90 degrees counterclockwise 03H Text is rotated 90 degrees clockwise

XF_ROTATION Text Rotation Angle (1 Byte), BIFF8 Value Text rotation 0 Not rotated 1-90 1 to 90 degrees counterclockwise 91-180 1 to 90 degrees clockwise 255 Letters are stacked top-to-bottom, but not rotated

XF_BORDER_34 Cell Border Style (4 Bytes), BIFF3-BIFF4 Cell borders contain a line style and a line colour for each line of the border. Bit Mask Contents 2-0 00000007H Top line style 7-3 000000F8H Colour index for top line colour 10-8 00000700H Left line style 15-11 0000F800H Colour index for left line colour 18-16 00070000H Bottom line style 23-19 00F80000H Colour index

for bottom line colour 26-24 07000000H Right line style 31-27 F8000000H Colour index for right line colour

XF_AREA_34 Cell Background Area Style (2 Bytes), BIFF3-BIFF4 A cell background area style contains an area pattern and a foreground and background colour. Bit Mask Contents 5-0 003FH Fill pattern 10-6 07C0H Colour index for pattern colour 15-11 F800H Colour index for pattern background

Record XF, BIFF8: Offset Size Contents 0 2 Index to FONT record 2 2 Index to FORMAT record 4 2 Bit Mask Contents

2-0 0007H XF_TYPE_PROT . XF type, cell protection (see above) 15-4 FFF0H Index to parent style XF (always FFFH in style XFs)

6 1 Bit Mask Contents 2-0 07H XF_HOR_ALIGN . Horizontal alignment (see above) 3 08H 1 = Text is wrapped at right border 6-4 70H XF_VERT_ALIGN . Vertical alignment (see above)

7 1 XF_ROTATION: Text rotation angle (see above) 8 1 Bit Mask Contents

3-0 0FH Indent level 4 10H 1 = Shrink content to fit into cell 5 merge 7-6 C0H Text direction (BIFF8X only)

00b = According to context 01b = Left-to-right 10b = Right-to-left

9 1 Bit Mask Contents 7-2 FCH XF_USED_ATTRIB . Used attributes (see above)

10 4 Cell border lines and background area: Bit Mask Contents 3-0 0000000FH Left line style 7-4 000000F0H Right line style 11-8 00000F00H Top line style 15-12 0000F000H Bottom line style 22-16 007F0000H Colour index for left line colour 29-23 3F800000H Colour index for right line colour 30 40000000H 1 = Diagonal line from top left to right bottom 31 80000000H 1 = Diagonal line from bottom left to right top

14 4 Bit Mask Contents 6-0 0000007FH Colour index for top line colour 13-7 00003F80H Colour index for bottom line colour 20-14 001FC000H Colour index for diagonal line colour 24-21 01E00000H Diagonal line style 31-26 FC000000H Fill pattern

18 2 Bit Mask Contents 6-0 007FH Colour index for pattern colour 13-7 3F80H Colour index for pattern background

class xlwt.BIFFRecords.**XcallSupBookRecord**
Bases: xlwt.BIFFRecords.SupBookRecord

Add-in function names are stored in EXTERNNAME records following this record.

Offset Size Contents 0 2 0001H 2 2 01H 3AH (relict of BIFF5, the byte string ':', see EXTERNSHEET record, 5.41)

4.2.3 Bitmap Module

class xlwt.Bitmap.**ImDataBmpRecord**(*filename*)
Bases: xlwt.BIFFRecords.BiffRecord

class xlwt.Bitmap.**ObjBmpRecord**(*row*, *col*, *sheet*, *im_data_bmp*, *x*, *y*, *scale_x*, *scale_y*)
Bases: xlwt.BIFFRecords.BiffRecord

4.2.4 `Cell` Module

class `xlwt.Cell.`**`BlankCell`**(*rowx*, *colx*, *xf_idx*)
Bases: `object`

`colx`

`get_biff_data`()

`rowx`

`xf_idx`

class `xlwt.Cell.`**`BooleanCell`**(*rowx*, *colx*, *xf_idx*, *number*)
Bases: `object`

`colx`

`get_biff_data`()

`number`

`rowx`

`xf_idx`

class `xlwt.Cell.`**`ErrorCell`**(*rowx*, *colx*, *xf_idx*, *error_string_or_code*)
Bases: `object`

`colx`

`get_biff_data`()

`number`

`rowx`

`xf_idx`

class `xlwt.Cell.`**`FormulaCell`**(*rowx*, *colx*, *xf_idx*, *frmla*, *calc_flags=0*)
Bases: `object`

`calc_flags`

`colx`

`frmla`

`get_biff_data`()

`rowx`

`xf_idx`

class `xlwt.Cell.`**`MulBlankCell`**(*rowx*, *colx1*, *colx2*, *xf_idx*)
Bases: `object`

`colx1`

`colx2`

`get_biff_data`()

`rowx`

`xf_idx`

class `xlwt.Cell.`**`NumberCell`**(*rowx*, *colx*, *xf_idx*, *number*)
Bases: `object`

colx

get_biff_data()

get_encoded_data()

number

rowx

xf_idx

class xlwt.Cell.**StrCell**(*rowx*, *colx*, *xf_idx*, *sst_idx*)
Bases: object

colx

get_biff_data()

rowx

sst_idx

xf_idx

4.2.5 Column Module

class xlwt.Column.**Column**(*colx*, *parent_sheet*)
Bases: object

get_biff_record()

set_style(*style*)

width_in_pixels()

4.2.6 CompoundDoc Module

class xlwt.CompoundDoc.**Reader**(*filename*, *dump=False*)

get_stream_data(*data*, *SAT*, *start_sid*, *sect_size*)

class xlwt.CompoundDoc.**XlsDoc**

MIN_LIMIT = 4096

SECTOR_SIZE = 512

SID_END_OF_CHAIN = -2

SID_FREE_SECTOR = -1

SID_USED_BY_MSAT = -4

SID_USED_BY_SAT = -3

save(*file_name_or_filelike_obj*, *stream*)

xlwt.CompoundDoc.**print_bin_data**(*data*)

4.2.7 ExcelFormula Module

class xlwt.ExcelFormula.**Formula**(*s*)
Bases: object

get_references()

patch_references(*patches*)

rpn()
Offset Size Contents 0 2 Size of the following formula data (sz) 2 sz Formula data (RPN token array) [2+sz] var. (optional) Additional data for specific tokens

text()

4.2.8 ExcelFormulaLexer Module

class xlwt.ExcelFormulaLexer.**Lexer**(*text*)
Bases: xlwt.antlr.TokenStream

curr_ch()

isEOF()

is_whitespace()

match_pattern()

nextToken()

next_ch(*n=1*)

4.2.9 ExcelFormulaParser Module

exception xlwt.ExcelFormulaParser.**FormulaParseException**
Bases: exceptions.Exception

An exception indicating that a Formula could not be successfully parsed.

class xlwt.ExcelFormulaParser.**Parser**(**args*, ***kwargs*)
Bases: xlwt.antlr.LLkParser

expr(*arg_type*)

expr_list(*arg_type_list*, *min_argc*, *max_argc*)

formula()

prec0_expr(*arg_type*)

prec1_expr(*arg_type*)

prec2_expr(*arg_type*)

prec3_expr(*arg_type*)

prec4_expr(*arg_type*)

prec5_expr(*arg_type*)

primary(*arg_type*)

sheet()

xlwt.ExcelFormulaParser.**mk_tokenSet_0**()

4.2.10 ExcelMagic Module

lots of Excel Magic Numbers

4.2.11 Formatting Module

The XF record is able to store explicit cell formatting attributes or the attributes of a cell style. Explicit formatting includes the reference to a cell style XF record. This allows to extend a defined cell style with some explicit attributes. The formatting attributes are divided into 6 groups:

Group Attributes

Number format Number format index (index to FORMAT record) Font Font index (index to FONT record) Alignment Horizontal and vertical alignment, text wrap, indentation,

orientation/rotation, text direction

Border Border line styles and colours Background Background area style and colours Protection Cell locked, formula hidden

For each group a flag in the cell XF record specifies whether to use the attributes contained in that XF record or in the referenced style XF record. In style XF records, these flags specify whether the attributes will overwrite explicit cell formatting when the style is applied to a cell. Changing a cell style (without applying this style to a cell) will change all cells which already use that style and do not contain explicit cell attributes for the changed style attributes. If a cell XF record does not contain explicit attributes in a group (if the attribute group flag is not set), it repeats the attributes of its style XF record.

class xlwt.Formatting.**Alignment**

Bases: object

DIRECTION_GENERAL = 0

DIRECTION_LR = 1

DIRECTION_RL = 2

HORZ_CENTER = 2

HORZ_CENTER_ACROSS_SEL = 6

HORZ_DISTRIBUTED = 7

HORZ_FILLED = 4

HORZ_GENERAL = 0

HORZ_JUSTIFIED = 5

HORZ_LEFT = 1

HORZ_RIGHT = 3

NOT_SHRINK_TO_FIT = 0

NOT_WRAP_AT_RIGHT = 0

ORIENTATION_90_CC = 2

ORIENTATION_90_CW = 3

ORIENTATION_NOT_ROTATED = 0

ORIENTATION_STACKED = 1

ROTATION_0_ANGLE = 0

ROTATION_STACKED = 255

SHRINK_TO_FIT = 1

VERT_BOTTOM = 2

VERT_CENTER = 1

VERT_DISTRIBUTED = 4

VERT_JUSTIFIED = 3

VERT_TOP = 0

WRAP_AT_RIGHT = 1

class xlwt.Formatting.**Borders**
Bases: object

DASHED = 3

DOTTED = 4

DOUBLE = 6

HAIR = 7

MEDIUM = 2

MEDIUM_DASHED = 8

MEDIUM_DASH_DOTTED = 10

MEDIUM_DASH_DOT_DOTTED = 12

NEED_DIAG1 = 1

NEED_DIAG2 = 1

NO_LINE = 0

NO_NEED_DIAG1 = 0

NO_NEED_DIAG2 = 0

SLANTED_MEDIUM_DASH_DOTTED = 13

THICK = 5

THIN = 1

THIN_DASH_DOTTED = 9

THIN_DASH_DOT_DOTTED = 11

class xlwt.Formatting.**Font**
Bases: object

CHARSET_ANSI_ARABIC = 178

CHARSET_ANSI_BALTIC = 186

CHARSET_ANSI_CHINESE_BIG5 = 136

CHARSET_ANSI_CHINESE_GBK = 134

CHARSET_ANSI_CYRILLIC = 204

CHARSET_ANSI_GREEK = 161

CHARSET_ANSI_HEBREW = 177

CHARSET_ANSI_JAP_SHIFT_JIS = 128

CHARSET_ANSI_KOR_HANGUL = 129

CHARSET_ANSI_KOR_JOHAB = 130

CHARSET_ANSI_LATIN = 0

CHARSET_ANSI_LATIN_II = 238

CHARSET_ANSI_THAI = 222

CHARSET_ANSI_TURKISH = 162

CHARSET_ANSI_VIETNAMESE = 163

CHARSET_APPLE_ROMAN = 77

CHARSET_OEM_LATIN_I = 255

CHARSET_SYMBOL = 2

CHARSET_SYS_DEFAULT = 1

ESCAPEMENT_NONE = 0

ESCAPEMENT_SUBSCRIPT = 2

ESCAPEMENT_SUPERSCRIPT = 1

FAMILY_DECORATIVE = 5

FAMILY_MODERN = 3

FAMILY_NONE = 0

FAMILY_ROMAN = 1

FAMILY_SCRIPT = 4

FAMILY_SWISS = 2

UNDERLINE_DOUBLE = 2

UNDERLINE_DOUBLE_ACC = 34

UNDERLINE_NONE = 0

UNDERLINE_SINGLE = 1

UNDERLINE_SINGLE_ACC = 33

get_biff_record()

class xlwt.Formatting.**Pattern**

Bases: object

NO_PATTERN = 0

SOLID_PATTERN = 1

class xlwt.Formatting.**Protection**

Bases: object

4.2.12 Row Module

class xlwt.Row.**Row**(*rowx*, *parent_sheet*)
Bases: object

collapse

get_cells_biff_data()

get_cells_count()

get_height_in_pixels()

get_index()

get_max_col()

get_min_col()

get_row_biff_data()

get_xf_index()

has_default_height

height

height_mismatch

hidden

insert_cell(*col_index*, *cell_obj*)

insert_mulcells(*colx1*, *colx2*, *cell_obj*)

level

set_cell_blank(*colx*, *style=<xlwt.Style.XFStyle object at 0x72f6550>*)

set_cell_boolean(*colx*, *value*, *style=<xlwt.Style.XFStyle object at 0x72f6550>*)

set_cell_date(*colx*, *datetime_obj*, *style=<xlwt.Style.XFStyle object at 0x72f6550>*)

set_cell_error(*colx*, *error_string_or_code*, *style=<xlwt.Style.XFStyle object at 0x72f6550>*)

set_cell_formula(*colx*, *formula*, *style=<xlwt.Style.XFStyle object at 0x72f6550>*, *calc_flags=0*)

set_cell_mulblanks(*first_colx*, *last_colx*, *style=<xlwt.Style.XFStyle object at 0x72f6550>*)

set_cell_number(*colx*, *number*, *style=<xlwt.Style.XFStyle object at 0x72f6550>*)

set_cell_text(*colx*, *value*, *style=<xlwt.Style.XFStyle object at 0x72f6550>*)

set_style(*style*)

space_above

space_below

write(*col*, *label*, *style=<xlwt.Style.XFStyle object at 0x72f6550>*)

write_blanks(*first_colx*, *last_colx*, *style=<xlwt.Style.XFStyle object at 0x72f6550>*)

4.2.13 `Style` Module

exception `xlwt.Style.`**`EasyXFAuthorError`**
Bases: `xlwt.Style.EasyXFException`

exception `xlwt.Style.`**`EasyXFCallerError`**
Bases: `xlwt.Style.EasyXFException`

exception `xlwt.Style.`**`EasyXFException`**
Bases: `exceptions.Exception`

class `xlwt.Style.`**`IntULim`**(*limit*)
Bases: `object`

class `xlwt.Style.`**`StyleCollection`**(*style_compression=0*)
Bases: `object`

`add`(*style*)

`get_biff_data`()

class `xlwt.Style.`**`XFStyle`**
Bases: `object`

`xlwt.Style.`**`any_str_func`**(*s*)

`xlwt.Style.`**`colour_index_func`**(*s*, *maxval=127*)

`xlwt.Style.`**`colour_index_func_15`**(*s*)

`xlwt.Style.`**`colour_index_func_7`**(*s*, *maxval=127*)

`xlwt.Style.`**`easyxf`**(*strg_to_parse=''*, *num_format_str=None*, *field_sep=','*, *line_sep=';'*, *intro_sep=':'*, *esc_char='\\'*, *debug=False*)

`xlwt.Style.`**`rotation_func`**(*s*)

4.2.14 `UnicodeUtils` Module

From BIFF8 on, strings are always stored using UTF-16LE text encoding. The character array is a sequence of 16-bit values4. Additionally it is possible to use a compressed format, which omits the high bytes of all characters, if they are all zero.

The following tables describe the standard format of the entire string, but in many records the strings differ from this format. This will be mentioned separately. It is possible (but not required) to store Rich-Text formatting information and Asian phonetic information inside a Unicode string. This results in four different ways to store a string. The character array is not zero-terminated.

The string consists of the character count (as usual an 8-bit value or a 16-bit value), option flags, the character array and optional formatting information. If the string is empty, sometimes the option flags field will not occur. This is mentioned at the respective place.

Offset Size Contents 0 1 or 2 Length of the string (character count, ln) 1 or 2 1 Option flags:

Bit Mask Contents 0 01H Character compression (ccompr):

0 = Compressed (8-bit characters) 1 = Uncompressed (16-bit characters)

2 04H Asian phonetic settings (phonetic): 0 = Does not contain Asian phonetic settings 1 = Contains Asian phonetic settings

3 08H Rich-Text settings (richtext): 0 = Does not contain Rich-Text settings 1 = Contains Rich-Text settings

[2 or 3] 2 (optional, only if richtext=1) Number of Rich-Text formatting runs (rt) [var.] 4 (optional, only if phonetic=1) Size of Asian phonetic settings block (in bytes, sz) var. ln or

2·ln Character array (8-bit characters or 16-bit characters, dependent on ccompr)

[var.] 4·rt (optional, only if richtext=1) List of rt formatting runs [var.] sz (optional, only if phonetic=1) Asian Phonetic Settings Block

xlwt.UnicodeUtils.**upack1**(*s*, *encoding='ascii'*)

xlwt.UnicodeUtils.**upack2**(*s*, *encoding='ascii'*)

4.2.15 `Utils` Module

xlwt.Utils.**cell_to_packed_rowcol**(*cell*)
pack row and column into the required 4 byte format

xlwt.Utils.**cell_to_rowcol**(*cell*)
Convert an Excel cell reference string in A1 notation to numeric row/col notation.

Returns: row, col, row_abs, col_abs

xlwt.Utils.**cell_to_rowcol2**(*cell*)
Convert an Excel cell reference string in A1 notation to numeric row/col notation.

Returns: row, col

xlwt.Utils.**cellrange_to_rowcol_pair**(*cellrange*)
Convert cell range string in A1 notation to numeric row/col pair.

Returns: row1, col1, row2, col2

xlwt.Utils.**col_by_name**(*colname*)

xlwt.Utils.**quote_sheet_name**(*unquoted_sheet_name*)

xlwt.Utils.**rowcol_pair_to_cellrange**(*row1*, *col1*, *row2*, *col2*, *row1_abs=False*, *col1_abs=False*, *row2_abs=False*, *col2_abs=False*)
Convert two (row,column) pairs into a cell range string in A1:B2 notation.

Returns: cell range string

xlwt.Utils.**rowcol_to_cell**(*row*, *col*, *row_abs=False*, *col_abs=False*)
Convert numeric row/col notation to an Excel cell reference string in A1 notation.

xlwt.Utils.**valid_sheet_name**(*sheet_name*)

4.2.16 `Workbook` Module

Record Order in BIFF8

Workbook Globals Substream

BOF Type = workbook globals Interface Header MMS Interface End WRITEACCESS CODEPAGE DSF TABID FNGROUPCOUNT Workbook Protection Block

WINDOWPROTECT PROTECT PASSWORD PROT4REV PROT4REVPASS

BACKUP HIDEOBJ WINDOW1 DATEMODE PRECISION REFRESHALL BOOKBOOL FONT + FORMAT * XF + STYLE +

? **PALETTE** USESELFS

BOUNDSHEET +

COUNTRY

? **Link Table** SST ExtSST EOF

class xlwt.Workbook.**Workbook**(*encoding='ascii'*, *style_compression=0*)

Bases: object

active_sheet

add_sheet(*sheetname*, *cell_overwrite_ok=False*)

add_sheet_reference(*formula*)

add_str(*s*)

add_style(*style*)

backup_on_save

convert_sheetindex(*strg_ref*, *n_sheets*)

country_code

dates_1904

default_style

del_str(*sst_idx*)

get_active_sheet()

get_backup_on_save()

get_biff_data()

get_country_code()

get_dates_1904()

get_default_style()

get_height()

get_hpos()

get_hscroll_visible()

get_obj_protect()

get_owner()

get_protect()

get_sheet(*sheetnum*)

get_style_stats()

get_tab_width()

get_tabs_visible()

get_use_cell_values()

get_vpos()

get_vscroll_visible()

get_width()

get_wnd_mini()

get_wnd_protect()

get_wnd_visible()

height

hpos

hscroll_visible

obj_protect

owner

protect

raise_bad_sheetname(*sheetname*)

save(*filename*)

set_active_sheet(*value*)

set_backup_on_save(*value*)

set_country_code(*value*)

set_dates_1904(*value*)

set_height(*value*)

set_hpos(*value*)

set_hscroll_visible(*value*)

set_obj_protect(*value*)

set_owner(*value*)

set_protect(*value*)

set_tab_width(*value*)

set_tabs_visible(*value*)

set_use_cell_values(*value*)

set_vpos(*value*)

set_vscroll_visible(*value*)

set_width(*value*)

set_wnd_mini(*value*)

set_wnd_protect(*value*)

set_wnd_visible(*value*)

setup_ownbook()

setup_xcall()

str_index(*s*)

tab_width

tabs_visible

use_cell_values

vpos

vscroll_visible

width

wnd_mini

wnd_protect

wnd_visible

4.2.17 `Worksheet` Module

BOF UNCALCED INDEX Calculation Settings Block PRINTHEADERS PRINTGRIDLINES GRIDSET GUTS DEFAULTROWHEIGHT WSBOOL Page Settings Block Worksheet Protection Block DEFCOLWIDTH COLINFO SORT DIMENSIONS Row Blocks WINDOW2 SCL PANE SELECTION STANDARDWIDTH MERGEDCELLS LABELRANGES PHONETIC Conditional Formatting Table Hyperlink Table Data Validity Table SHEETLAYOUT (BIFF8X only) SHEETPROTECTION (BIFF8X only) RANGEPROTECTION (BIFF8X only) EOF

class `xlwt.Worksheet.`**Worksheet**(*sheetname*, *parent_book*, *cell_overwrite_ok=False*)

Bases: `object`

RC_ref_mode

class **Workbook**(*encoding='ascii'*, *style_compression=0*)

Bases: `object`

active_sheet

add_sheet(*sheetname*, *cell_overwrite_ok=False*)

add_sheet_reference(*formula*)

add_str(*s*)

add_style(*style*)

backup_on_save

convert_sheetindex(*strg_ref*, *n_sheets*)

country_code

dates_1904

default_style

del_str(*sst_idx*)

get_active_sheet()

get_backup_on_save()

get_biff_data()

get_country_code()

get_dates_1904()

get_default_style()

get_height()

get_hpos()

get_hscroll_visible()

get_obj_protect()

get_owner()

get_protect()

get_sheet(*sheetnum*)

get_style_stats()

get_tab_width()

get_tabs_visible()

get_use_cell_values()

get_vpos()

get_vscroll_visible()

get_width()

get_wnd_mini()

get_wnd_protect()

get_wnd_visible()

height

hpos

hscroll_visible

obj_protect

owner

protect

raise_bad_sheetname(*sheetname*)

save(*filename*)

set_active_sheet(*value*)

set_backup_on_save(*value*)

set_country_code(*value*)

set_dates_1904(*value*)

set_height(*value*)

set_hpos(*value*)

set_hscroll_visible(*value*)

set_obj_protect(*value*)

set_owner(*value*)

set_protect(*value*)

set_tab_width(*value*)

set_tabs_visible(*value*)

set_use_cell_values(*value*)

set_vpos(*value*)

set_vscroll_visible(*value*)

set_width(*value*)

set_wnd_mini(*value*)

set_wnd_protect(*value*)

set_wnd_visible(*value*)

setup_ownbook()

setup_xcall()

str_index(*s*)

tab_width

tabs_visible

use_cell_values

vpos

vscroll_visible

width

wnd_mini

wnd_protect

wnd_visible

Worksheet.**alt_expr_eval**

Worksheet.**alt_formula_entries**

Worksheet.**auto_colour_grid**

Worksheet.**auto_style_outline**

Worksheet.**bmp_rec**

Worksheet.**bottom_margin**

Worksheet.**calc_count**

Worksheet.**calc_mode**

Worksheet.**col**(*indx*)

Worksheet.**col_default_width**

Worksheet.**col_width**(*col*)

Worksheet.**cols**

Worksheet.**cols_right_to_left**

Worksheet.**copies_num**

Worksheet.**delta**

Worksheet.**dialogue_sheet**

Worksheet.**first_visible_col**

Worksheet.**first_visible_row**

Worksheet.**fit_height_to_pages**

Worksheet.**fit_num_pages**

Worksheet.**fit_width_to_pages**

Worksheet.**flush_row_data**()

Worksheet.**footer_margin**

Worksheet.**footer_str**

Worksheet.**get_RC_ref_mode**()

Worksheet.**get_alt_expr_eval**()

Worksheet.**get_alt_formula_entries**()

Worksheet.**get_auto_colour_grid**()

Worksheet.**get_auto_style_outline**()

Worksheet.**get_biff_data**()

Worksheet.**get_bmp_rec**()

Worksheet.**get_bottom_margin**()

Worksheet.**get_calc_count**()

Worksheet.**get_calc_mode**()

Worksheet.**get_col_default_width**()

Worksheet.**get_cols**()

Worksheet.**get_cols_right_to_left**()

Worksheet.**get_copies_num**()

Worksheet.**get_delta**()

Worksheet.**get_dialogue_sheet**()

Worksheet.**get_first_visible_col**()

Worksheet.**get_first_visible_row**()

Worksheet.**get_fit_height_to_pages**()

Worksheet.**get_fit_num_pages**()

Worksheet.**get_fit_width_to_pages**()

Worksheet.**get_footer_margin**()

Worksheet.**get_footer_str**()

Worksheet.**get_grid_colour**()

Worksheet.**get_header_margin**()

Worksheet.**get_header_str**()

Worksheet.**get_horz_page_breaks**()

Worksheet.**get_horz_split_first_visible**()

Worksheet.**get_horz_split_pos**()

Worksheet.**get_iterations_on**()

Worksheet.**get_left_margin**()

Worksheet.**get_merged_ranges**()

Worksheet.**get_name**()

Worksheet.**get_normal_magn**()

Worksheet.**get_obj_protect**()

Worksheet.**get_outline_below**()

Worksheet.**get_outline_right**()

Worksheet.**get_page_preview**()

Worksheet.**get_panes_frozen**()

Worksheet.**get_paper_size_code**()

Worksheet.**get_parent**()

Worksheet.**get_password**()

Worksheet.**get_portrait**()

Worksheet.**get_preview_magn**()

Worksheet.**get_print_centered_horz**()

Worksheet.**get_print_centered_vert**()

Worksheet.**get_print_colour**()

Worksheet.**get_print_draft**()

Worksheet.**get_print_grid**()

Worksheet.**get_print_headers**()

Worksheet.**get_print_hres**()

Worksheet.**get_print_in_rows**()

Worksheet.**get_print_notes**()

Worksheet.**get_print_notes_at_end**()

Worksheet.**get_print_omit_errors**()

Worksheet.**get_print_scaling**()

Worksheet.**get_print_vres**()

Worksheet.**get_protect**()

Worksheet.**get_remove_splits**()

Worksheet.**get_right_margin**()

Worksheet.**get_row_default_height**()

Worksheet.**get_rows**()

Worksheet.**get_save_recalc**()

Worksheet.**get_scen_protect**()

Worksheet.**get_selected**()

Worksheet.**get_sheet_visible**()

Worksheet.**get_show_auto_page_breaks**()

Worksheet.**get_show_col_outline**()

Worksheet.**get_show_formulas**()

Worksheet.**get_show_grid**()

Worksheet.**get_show_headers**()

Worksheet.**get_show_outline**()

Worksheet.**get_show_row_outline**()

Worksheet.**get_start_page_number**()

Worksheet.**get_top_margin**()

Worksheet.**get_vert_page_breaks**()

Worksheet.**get_vert_split_first_visible**()

Worksheet.**get_vert_split_pos**()

Worksheet.**get_wnd_protect**()

Worksheet.**grid_colour**

Worksheet.**header_margin**

Worksheet.**header_str**

Worksheet.**horz_page_breaks**

Worksheet.**horz_split_first_visible**

Worksheet.**horz_split_pos**

Worksheet.**insert_bitmap**(*filename*, *row*, *col*, *x=0*, *y=0*, *scale_x=1*, *scale_y=1*)

Worksheet.**iterations_on**

Worksheet.**left_margin**

Worksheet.**merge**(*r1*, *r2*, *c1*, *c2*, *style=<xlwt.Style.XFStyle object at 0x72f6550>*)

Worksheet.**merged_ranges**

Worksheet.**name**

Worksheet.**normal_magn**

Worksheet.**obj_protect**

Worksheet.**outline_below**

Worksheet.**outline_right**

Worksheet.**page_preview**

Worksheet.**panes_frozen**

Worksheet.**paper_size_code**

Worksheet.**parent**

Worksheet.**password**

Worksheet.**portrait**

Worksheet.**preview_magn**

Worksheet.**print_centered_horz**

Worksheet.**print_centered_vert**

Worksheet.**print_colour**

Worksheet.**print_draft**

Worksheet.**print_grid**

Worksheet.**print_headers**

Worksheet.**print_hres**

Worksheet.**print_in_rows**

Worksheet.**print_notes**

Worksheet.**print_notes_at_end**

Worksheet.**print_omit_errors**

Worksheet.**print_scaling**

Worksheet.**print_vres**

Worksheet.**protect**

Worksheet.**remove_splits**

Worksheet.**right_margin**

Worksheet.**row**(*indx*)

Worksheet.**row_default_height**

Worksheet.**row_height**(*row*)

Worksheet.**rows**

Worksheet.**save_recalc**

Worksheet.**scen_protect**

Worksheet.**selected**

Worksheet.**set_RC_ref_mode**(*value*)

Worksheet.**set_alt_expr_eval**(*value*)

Worksheet.**set_alt_formula_entries**(*value*)

Worksheet.**set_auto_colour_grid**(*value*)

Worksheet.**set_auto_style_outline**(*value*)

Worksheet.**set_bottom_margin**(*value*)

Worksheet.**set_calc_count**(*value*)

Worksheet.**set_calc_mode**(*value*)

Worksheet.**set_col_default_width**(*value*)

Worksheet.**set_cols_right_to_left**(*value*)

Worksheet.**set_copies_num**(*value*)

Worksheet.**set_delta**(*value*)

Worksheet.**set_dialogue_sheet**(*value*)

Worksheet.**set_first_visible_col**(*value*)

Worksheet.**set_first_visible_row**(*value*)

Worksheet.**set_fit_height_to_pages**(*value*)

Worksheet.**set_fit_num_pages**(*value*)

Worksheet.**set_fit_width_to_pages**(*value*)

Worksheet.**set_footer_margin**(*value*)

Worksheet.**set_footer_str**(*value*)

Worksheet.**set_grid_colour**(*value*)

Worksheet.**set_header_margin**(*value*)

Worksheet.**set_header_str**(*value*)

Worksheet.**set_horz_page_breaks**(*value*)

Worksheet.**set_horz_split_first_visible**(*value*)

Worksheet.**set_horz_split_pos**(*value*)

Worksheet.**set_iterations_on**(*value*)

Worksheet.**set_left_margin**(*value*)

Worksheet.**set_name**(*value*)

Worksheet.**set_normal_magn**(*value*)

Worksheet.**set_obj_protect**(*value*)

Worksheet.**set_outline_below**(*value*)

Worksheet.**set_outline_right**(*value*)

Worksheet.**set_page_preview**(*value*)

Worksheet.**set_panes_frozen**(*value*)

Worksheet.**set_paper_size_code**(*value*)

Worksheet.**set_password**(*value*)

Worksheet.**set_portrait**(*value*)

Worksheet.**set_preview_magn**(*value*)

Worksheet.**set_print_centered_horz**(*value*)

Worksheet.**set_print_centered_vert**(*value*)

Worksheet.**set_print_colour**(*value*)

Worksheet.**set_print_draft**(*value*)

Worksheet.**set_print_grid**(*value*)

Worksheet.**set_print_headers**(*value*)

Worksheet.**set_print_hres**(*value*)

Worksheet.**set_print_in_rows**(*value*)

Worksheet.**set_print_notes**(*value*)

Worksheet.**set_print_notes_at_end**(*value*)

Worksheet.**set_print_omit_errors**(*value*)

Worksheet.**set_print_scaling**(*value*)

Worksheet.**set_print_vres**(*value*)

Worksheet.**set_protect**(*value*)

Worksheet.**set_remove_splits**(*value*)

Worksheet.**set_right_margin**(*value*)

Worksheet.**set_row_default_height**(*value*)

Worksheet.**set_save_recalc**(*value*)

Worksheet.**set_scen_protect**(*value*)

Worksheet.**set_selected**(*value*)

Worksheet.**set_sheet_visible**(*value*)

Worksheet.**set_show_auto_page_breaks**(*value*)

Worksheet.**set_show_col_outline**(*value*)

Worksheet.**set_show_formulas**(*value*)

Worksheet.**set_show_grid**(*value*)

Worksheet.**set_show_headers**(*value*)

Worksheet.**set_show_outline**(*value*)

Worksheet.**set_show_row_outline**(*value*)

Worksheet.**set_start_page_number**(*value*)

Worksheet.**set_top_margin**(*value*)

Worksheet.**set_vert_page_breaks**(*value*)

Worksheet.**set_vert_split_first_visible**(*value*)

Worksheet.**set_vert_split_pos**(*value*)

Worksheet.**set_wnd_protect**(*value*)

Worksheet.**sheet_visible**

Worksheet.**show_auto_page_breaks**

Worksheet.**show_col_outline**

Worksheet.**show_formulas**

Worksheet.**show_grid**

Worksheet.**show_headers**

Worksheet.**show_outline**

Worksheet.**show_row_outline**

Worksheet.**start_page_number**

Worksheet.**top_margin**

Worksheet.**vert_page_breaks**

Worksheet.**vert_split_first_visible**

Worksheet.**vert_split_pos**

Worksheet.**wnd_protect**

Worksheet.**write**(*r*, *c*, *label=''*, *style=<xlwt.Style.XFStyle object at 0x72f6550>*)

Worksheet.**write_merge**(*r1*, *r2*, *c1*, *c2*, *label=''*, *style=<xlwt.Style.XFStyle object at 0x72f6550>*)

4.2.18 antlr Module

exception xlwt.antlr.**ANTLRException**(**args*)
Bases: exceptions.Exception

class xlwt.antlr.**AST**
Bases: object

addChild(*c*)

equals(*t*)

equalsList(*t*)

equalsListPartial(*t*)

equalsTree(*t*)

equalsTreePartial(*t*)

findAll(*tree*)

findAllPartial(*subtree*)

getColumn()

getFirstChild()

getLine()

getNextSibling()

getNumberOfChildren()

getText()

getType()

initialize(*t*)

setFirstChild(*c*)

setNextSibling(*n*)

setText(*text*)

setType(*ttype*)

toString()

toStringList()

toStringTree()

class xlwt.antlr.**ASTFactory**(*table=None*)
Bases: object

create(**args*)

dup(*t*)

dupList(*t*)

dupTree(*t*)

error(*e*)

getASTNodeClass()

getASTNodeType(*tokenType*)
For a given token type return the AST node type. First we lookup a mapping table, second we try _class and finally we resolve to "antlr.CommonAST".

getTokenTypeToASTClassMap()

maptype(*tokenType*, *className*)
Specify a mapping between a token type and a (AST) class.

setASTNodeClass(*className=None*)

setASTNodeType(*className=None*)

setTokenTypeASTNodeType(*tokenType*, *className*)
Specify a mapping between a token type and a (AST) class.

setTokenTypeToASTClassMap(*amap*)

class xlwt.antlr.**ASTNULLType**
Bases: xlwt.antlr.AST

getText()

getType()

class xlwt.antlr.**ASTPair**
Bases: object

advanceChildToEnd()

copy()

toString()

class xlwt.antlr.**ASTVisitor**(**args*)
Bases: object

visit(*ast*)

class xlwt.antlr.**BaseAST**
Bases: xlwt.antlr.AST

addChild(*node*)

doWorkForFindAll(*v*, *target*, *partialMatch*)

equals(*t*)

equalsList(*t*)

equalsListPartial(*t*)

equalsTree(*t*)

equalsTreePartial(*t*)

findAll(*target*)

findAllPartial(*sub*)

getColumn()

getFirstChild()

getLine()

getNextSibling()

getNumberOfChildren()

getText()

getTokenNames()

getType()

removeChildren()

setFirstChild(*c*)

setNextSibling(*n*)

setText(*text*)

setType(*ttype*)

static **setVerboseStringConversion**(*verbose*, *names*)

toString()

toStringList()

toStringTree()

tokenNames = None

verboseStringConversion = False

class xlwt.antlr.**BitSet**(*data=None*)
Bases: object

BITS = 64

LOG_BITS = 6

MOD_MASK = 63

NIBBLE = 4

add(*bit*, *on=True*)

at(*bit*)

bitMask(*bit*)

member(*item*)

off(*bit*, *off=True*)

set(*bit*, *on=True*)

wordNumber(*bit*)

class xlwt.antlr.**CharBuffer**(*reader*)
Bases: xlwt.antlr.InputBuffer

fill(*amount*)

class xlwt.antlr.**CharScanner**(**argv*, ***kwargs*)
Bases: xlwt.antlr.TokenStream

EOF_CHAR = ''

LA(*i*)

NO_CHAR = **0**

append(*c*)

commit()

consume()

consumeUntil_bitset(*bitset*)

consumeUntil_char(*c*)

default(*la1*)

filterdefault(*la1*, **args*)

getCaseSensitive()

getCaseSensitiveLiterals()

getColumn()

getCommitToPath()

getFilename()

getInputBuffer()

getInputState()

getLine()

getTabSize()

getText()

getTokenObject()

makeToken(*type*)

mark()

match(*item*)

matchNot(*c*)

matchRange(*c1*, *c2*)

newline()

panic(*s=''*)

raise_NoViableAlt(*la1=None*)

reportError(*s*)

reportWarning(*s*)

resetText()

rewind(*pos*)

setCaseSensitive(*t*)

setColumn(*c*)

setCommitToPath(*commit*)

setFilename(*f*)

setInput(**argv*)

setInputState(*state*)

setLine(*line*)

setTabSize(*size*)

setText(*s*)

setTokenObjectClass(*cl*)

set_return_token(*_create*, *_token*, *_ttype*, *_offset*)

tab()

testForLiteral(*token*)

testLiteralsTable(**args*)

toLower(*c*)

traceIn(*rname*)

traceIndent()

traceOut(*rname*)

uponEOF()

class xlwt.antlr.**CharScannerIterator**(*inst*)

next()

exception xlwt.antlr.**CharStreamException**(**args*)
Bases: xlwt.antlr.ANTLRException

exception xlwt.antlr.**CharStreamIOException**(**args*)
Bases: xlwt.antlr.CharStreamException

class xlwt.antlr.**CommonAST**(*token=None*)
Bases: xlwt.antlr.BaseAST

getColumn()

getLine()

getText()

getType()

initialize(**args*)

setText(*text_*)

setType(*ttype_*)

class xlwt.antlr.**CommonASTWithHiddenTokens**(**args*)
Bases: xlwt.antlr.CommonAST

getHiddenAfter()

getHiddenBefore()

initialize(**args*)

class xlwt.antlr.**CommonHiddenStreamToken**(**args*)
Bases: xlwt.antlr.CommonToken

getHiddenAfter()

getHiddenBefore()

setHiddenAfter(*t*)

setHiddenBefore(*t*)

class xlwt.antlr.**CommonToken**(***argv*)
Bases: xlwt.antlr.Token

getColumn()

getLine()

getText()

setColumn(*col*)

setLine(*line*)

setText(*text*)

toString()

class xlwt.antlr.**InputBuffer**
Bases: object

LA(*k*)

commit()

consume()

fill(*k*)

getLAChars()

getMarkedChars()

isMarked()

mark()

reset()

rewind(*mark*)

syncConsume()

class xlwt.antlr.**LLkParser**(**args*, ***kwargs*)
Bases: xlwt.antlr.Parser

LA(*i*)

LT(*i*)

consume()

set_k(*index*, **args*)

trace(*ee*, *rname*)

traceIn(*rname*)

traceOut(*rname*)

class xlwt.antlr.**LexerSharedInputState**(*ibuf*)
Bases: object

LA(*k*)

reset()

exception xlwt.antlr.**MismatchedCharException**(**args*)
Bases: xlwt.antlr.RecognitionException

CHAR = 1

NONE = 0

NOT_CHAR = 2

NOT_RANGE = 4

NOT_SET = 6

RANGE = 3

SET = 5

appendCharName(*sb*, *c*)

exception xlwt.antlr.**MismatchedTokenException**(**args*)
Bases: xlwt.antlr.RecognitionException

NONE = 0

NOT_RANGE = 4

NOT_SET = 6

NOT_TOKEN = 2

RANGE = 3

SET = 5

TOKEN = 1

appendTokenName(*sb*, *tokenType*)

exception xlwt.antlr.**NoViableAltException**(**args*)
Bases: xlwt.antlr.RecognitionException

exception xlwt.antlr.**NoViableAltForCharException**(**args*)
Bases: xlwt.antlr.RecognitionException

class xlwt.antlr.**Parser**(**args*, ***kwargs*)
Bases: object

LA(*i*)

LT(*i*)

addASTChild(*currentAST*, *child*)

addMessageListener(*l*)

addParserListener(*l*)

addParserMatchListener(*l*)

addParserTokenListener(*l*)

addSemanticPredicateListener(*l*)

addSyntacticPredicateListener(*l*)

addTraceListener(*l*)

consume()

consumeUntil(*arg*)

defaultDebuggingSetup()

getAST()

getASTFactory()

getFilename()

getInputState()

getTokenName(*num*)

getTokenNames()

getTokenTypeToASTClassMap()

isDebugMode()

makeASTRoot(*currentAST*, *root*)

mark()

match(*set*)

matchNot(*t*)

removeMessageListener(*l*)

removeParserListener(*l*)

removeParserMatchListener(*l*)

removeParserTokenListener(*l*)

removeSemanticPredicateListener(*l*)

removeSyntacticPredicateListener(*l*)

removeTraceListener(*l*)

reportError(*x*)

reportWarning(*s*)

rewind(*pos*)

setASTFactory(*f*)

setASTNodeClass(*cl*)

setASTNodeType(*nodeType*)

setDebugMode(*debugMode*)

setFilename(*f*)

setIgnoreInvalidDebugCalls(*value*)

setInputState(*state*)

setTokenBuffer(*t*)

traceIn(*rname*)

traceIndent()

traceOut(*rname*)

class xlwt.antlr.**ParserSharedInputState**
Bases: object

reset()

class xlwt.antlr.**Queue**
Bases: object

append(*item*)

elementAt(*index*)

length()

removeFirst()

reset()

class xlwt.antlr.**Reader**(*stream*)
Bases: object

read(*num*)

exception xlwt.antlr.**RecognitionException**(**args*)
Bases: xlwt.antlr.ANTLRException

exception xlwt.antlr.**SemanticException**(**args*)
Bases: xlwt.antlr.RecognitionException

class xlwt.antlr.**StringBuffer**(*string=None*)

append(*c*)

getString(*a=None*, *length=None*)

length()

setLength(*sz*)

toString(*a=None*, *length=None*)

class xlwt.antlr.**Token**(***argv*)
Bases: object

EOF = 1

EOF_TYPE = 1

INVALID_TYPE = 0

MIN_USER_TYPE = 4

NULL_TREE_LOOKAHEAD = 3

SKIP = -1

badToken = [”<no text>”,<INVALID_TYPE>]

getColumn()

getFilename()

getLine()

getText()

getType()

isEOF()

setColumn(*column*)

setFilename(*name*)

setLine(*line*)

setText(*text*)

setType(*type*)

toString()

class xlwt.antlr.**TokenBuffer**(*stream*)
Bases: object

LA(*k*)

LT(*k*)

consume()

fill(*amount*)

getInput()

mark()

reset()

rewind(*mark*)

syncConsume()

class xlwt.antlr.**TokenStream**
Bases: object

nextToken()

class xlwt.antlr.**TokenStreamBasicFilter**(*input*)
Bases: xlwt.antlr.TokenStream

discard(*arg*)

nextToken()

exception xlwt.antlr.**TokenStreamException**(**args*)
Bases: xlwt.antlr.ANTLRException

class xlwt.antlr.**TokenStreamHiddenTokenFilter**(*input*)
Bases: xlwt.antlr.TokenStreamBasicFilter

LA(*i*)

consume()

consumeFirst()

getDiscardMask()

getHiddenAfter(*t*)

getHiddenBefore(*t*)

getHideMask()

getInitialHiddenToken()

hide(*m*)

nextToken()

exception xlwt.antlr.**TokenStreamIOException**(**args*)
Bases: xlwt.antlr.TokenStreamException

class xlwt.antlr.**TokenStreamIterator**(*inst*)
Bases: object

next()

exception xlwt.antlr.**TokenStreamRecognitionException**(**args*)
Bases: xlwt.antlr.TokenStreamException

exception xlwt.antlr.**TokenStreamRetryException**(**args*)
Bases: xlwt.antlr.TokenStreamException

class xlwt.antlr.**TokenStreamSelector**
Bases: xlwt.antlr.TokenStream

addInputStream(*stream*, *key*)

getCurrentStream()

getStream(*sname*)

nextToken()

pop()

push(*arg*)

retry()

select(*arg*)

class xlwt.antlr.**TreeParser**(**args*, ***kwargs*)
Bases: object

addASTChild(*currentAST*, *child*)

getAST()

getASTFactory()

getTokenName(*num*)

getTokenNames()

makeASTRoot(*currentAST*, *root*)

match(*t*, *set*)

matchNot(*t*, *ttype*)

reportError(*ex*)

reportWarning(*s*)

setASTFactory(*f*)

setASTNodeClass(*nodeType*)

setASTNodeType(*nodeType*)

traceIn(*rname*, *t*)

traceIndent()

traceOut(*rname*, *t*)

class xlwt.antlr.**TreeParserSharedInputState**
Bases: object

exception xlwt.antlr.**TryAgain**
Bases: exceptions.Exception

xlwt.antlr.**assert_string_type**(*x*)

xlwt.antlr.**cmptree**(*s*, *t*, *partial*)

xlwt.antlr.**dup**(*t*, *factory*)

xlwt.antlr.**dupList**(*t*, *factory*)

xlwt.antlr.**dupTree**(*t*, *factory*)

xlwt.antlr.**error**(*fmt*, **args*)

xlwt.antlr.**ifelse**(*cond*, *_then*, *_else*)

xlwt.antlr.**illegalarg_ex**(*func*)

xlwt.antlr.**is_string_type**(*x*)

xlwt.antlr.**make**(**nodes*)

xlwt.antlr.**rightmost**(*ast*)

xlwt.antlr.**runtime_ex**(*func*)

xlwt.antlr.**version**()

4.2.19 licences Module

Portions copyright © 2007, Stephen John Machin, Lingfo Pty Ltd All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. None of the names of Stephen John Machin, Lingfo Pty Ltd and any contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

CHAPTER

FIVE

EXTERNAL MODULES PACKAGED WITH OPEN-HEA (USED FOR MYSQL DATABASE OPERATIONS) :

5.1 mysql Package

5.1.1 Subpackages

connector Package

connector Package

MySQL Connector/Python - MySQL drive written in Python

class mysql.connector.**MySQL**(**args*, ***kwargs*)
Bases: mysql.connector.mysql.MySQLBase

Class implementing Python DB API v2.0.

close()

commit()
Shortcut for executing COMMIT.

connect(*dsn=''*, *user=''*, *password=''*, *host='127.0.0.1'*, *port=3306*, *db=None*, *database=None*, *use_unicode=True*, *charset='utf8'*, *get_warnings=False*, *raise_on_warnings=False*, *autocommit=False*, *unix_socket=None*, *connection_timeout=None*, *client_flags=0*, *buffered=False*)

Establishes a connection to the MySQL Server. Called also when instansiating a new MySQLConnection object through the __init__ method.

Possible parameters are:

dsn (not used)

user The username used to authenticate with the MySQL Server.

password The password to authenticate the user with the MySQL Server.

host The hostname or the IP address of the MySQL Server we are connecting with. (default 127.0.0.1)

port TCP port to use for connecting to the MySQL Server. (default 3306)

database db

Initial database to use once we are connected with the MySQL Server. The db argument is synonym, but database takes precedence.

use_unicode If set to true, string values received from MySQL will be returned as Unicode strings. Default: True

charset Which character shall we use for sending data to MySQL. One can still override this by using the SET NAMES command directly, but this is discouraged. Instead, use the set_charset() method if you want to change it. Default: Whatever the MySQL server has default.

get_warnings If set to true, whenever a query gives a warning, a SHOW WARNINGS will be done to fetch them. They will be available as MySQLCursor.warnings. The default is to ignore these warnings, for debugging it's good to enable it though, or use strict mode in MySQL to make most of these warnings errors. Default: False

raise_on_warnings If set to True, warnings will be raised as exceptions. raise_on_warings overrides get_warnings. Default: False

autocommit Auto commit is OFF by default, which is required by the Python Db API 2.0 specification. Default: False

unix_socket Full path to the MySQL Server UNIX socket. By default TCP connection will be used using the address specified by the host argument.

connection_timeout Timeout for the TCP and UNIX socket connection.

client_flags

Allows to set flags for the connection. Check following for possible flags:

```
>>> from mysql.connector.constants import ClientFlag
>>> print '
```

'.join(ClientFlag.get_full_info())

buffered When set to True .execute() will fetch the rows immediatly.

cursor()

remove_cursor(*c*)

rollback()
Shortcut for executing ROLLBACK

mysql.connector.**Connect**(**args*, ***kwargs*)
Shortcut for creating a mysql.MySQL object.

class mysql.connector.**FieldType**
Bases: mysql.connector.constants._constants

BIT = 16

BLOB = 252

DATE = 10

DATETIME = 12

DECIMAL = 0

DOUBLE = 5

ENUM = 247

FLOAT = 4

GEOMETRY = 255

INT24 = 9

LONG = 3

LONGLONG = 8

LONG_BLOB = 251

MEDIUM_BLOB = 250

NEWDATE = 14

NEWDECIMAL = 246

NULL = 6

SET = 248

SHORT = 2

STRING = 254

TIME = 11

TIMESTAMP = 7

TINY = 1

TINY_BLOB = 249

VARCHAR = 15

VAR_STRING = 253

YEAR = 13

desc = {'LONGLONG': (8, 'LONGLONG'), 'SHORT': (2, 'SHORT'), 'LONG': (3, 'LONG'), 'DATE': (10, 'DATE'), 'N

classmethod get_binary_types()

classmethod get_number_types()

classmethod get_string_types()

classmethod get_timestamp_types()

prefix = 'FIELD_TYPE_'

class mysql.connector.**FieldFlag**

Bases: mysql.connector.constants._constantflags

Field flags as found in MySQL sources mysql-src/include/mysql_com.h

AUTO_INCREMENT = 512

BINARY = 128

BINCMP = 131072

BLOB = 16

ENUM = 256

FIELD_IN_ADD_INDEX = 1048576

FIELD_IN_PART_FUNC = 524288

FIELD_IS_RENAMED = 2097152

GET_FIXED_FIELDS = 262144

GROUP = 16384

MULTIPLE_KEY = 8

NOT_NULL = 1

NO_DEFAULT_VALUE = 4096

NUM = 16384

ON_UPDATE_NOW = 8192

PART_KEY = 32768

PRI_KEY = 2

SET = 2048

TIMESTAMP = 1024

UNIQUE = 65536

UNIQUE_KEY = 4

UNSIGNED = 32

ZEROFILL = 64

desc = {'GROUP': (16384, 'Intern: Group field'), 'FIELD_IN_PART_FUNC': (524288, 'Field part of partition func'), 'P

class mysql.connector.**ClientFlag**

Bases: mysql.connector.constants._constantflags

Client Options as found in the MySQL sources mysql-src/include/mysql_com.h

COMPRESS = 32

CONNECT_WITH_DB = 8

FOUND_ROWS = 2

IGNORE_SIGPIPE = 4096

IGNORE_SPACE = 256

INTERACTIVE = 1024

LOCAL_FILES = 128

LONG_FLAG = 4

LONG_PASSWD = 1

MULTI_RESULTS = 131072

MULTI_STATEMENTS = 65536

NO_SCHEMA = 16

ODBC = 64

PROTOCOL_41 = 512

REMEMBER_OPTIONS = 2147483648

RESERVED = 16384

SECURE_CONNECTION = 32768

SSL = 2048

SSL_VERIFY_SERVER_CERT = 1073741824

TRANSACTIONS = 8192

default = [1, 4, 8, 512, 8192, 32768, 65536, 131072]

desc = {'NO_SCHEMA': (16, "Don't allow database.table.column"), 'IGNORE_SIGPIPE': (4096, 'IGNORE sigpipes'),

classmethod get_default()

class mysql.connector.**CharacterSet**
Bases: mysql.connector.constants._constants

List of supported character sets with their collations. This maps to the character set we get from the server within the handshake packet.

To update this list, use the following query:

SELECT ID,CHARACTER_SET_NAME, COLLATION_NAME FROM INFORMATION_SCHEMA.COLLATIONS ORDER BY ID

This list is hardcoded because we want to avoid doing each time the above query to get the name of the character set used.

classmethod get_charset_info(*name*, *collation=None*)
Returns information about the charset and optional collation.

classmethod get_desc(*setid*)
Returns info string about the charset for given MySQL ID.

classmethod get_info(*setid*)
Returns information about the charset for given MySQL ID.

classmethod get_supported()
Returns a list with names of all supproted character sets.

class mysql.connector.**RefreshOption**
Bases: mysql.connector.constants._constants

Options used when sending the COM_REFRESH server command.

GRANT = 1

HOST = 8

LOG = 2

SLAVE = 64

STATUS = 16

TABLES = 4

THREADS = 32

desc = {'STATUS': (16, 'Flush status variables'), 'TABLES': (4, 'close all tables'), 'SLAVE': (64, 'Reset master info and

exception mysql.connector.**Error**(*m*, *errno=None*, *values=None*)
Bases: exceptions.StandardError

exception mysql.connector.**Warning**
Bases: exceptions.StandardError

exception mysql.connector.**InterfaceError**(*m=None*, *errno=None*, *values=None*)
Bases: mysql.connector.errors.Error

exception mysql.connector.**DatabaseError**(*m=None*, *errno=None*, *values=None*)
Bases: mysql.connector.errors.Error

exception mysql.connector.**NotSupportedError**(*m=None*, *errno=None*, *values=None*)
Bases: mysql.connector.errors.DatabaseError

exception mysql.connector.**DataError**(*m=None*, *errno=None*, *values=None*)
Bases: mysql.connector.errors.DatabaseError

exception mysql.connector.**IntegrityError**(*m=None*, *errno=None*, *values=None*)
Bases: mysql.connector.errors.DatabaseError

exception mysql.connector.**ProgrammingError**(*m=None*, *errno=None*, *values=None*)
Bases: mysql.connector.errors.DatabaseError

exception mysql.connector.**OperationalError**(*m=None*, *errno=None*, *values=None*)
Bases: mysql.connector.errors.DatabaseError

exception mysql.connector.**InternalError**(*m=None*, *errno=None*, *values=None*)
Bases: mysql.connector.errors.DatabaseError

mysql.connector.**connect**(**args*, ***kwargs*)
Shortcut for creating a mysql.MySQL object.

mysql.connector.**Date**
alias of date

mysql.connector.**Time**
alias of time

mysql.connector.**Timestamp**
alias of datetime

mysql.connector.**Binary**
alias of str

mysql.connector.**DateFromTicks**(*ticks*)

mysql.connector.**DateFromTicks**(*ticks*)

mysql.connector.**TimestampFromTicks**(*ticks*)

_version Module

Holds version of MySQL Connector/Python

connection Module

Implementing communication to MySQL servers

class mysql.connector.connection.**MySQLBaseConnection**(*prtcls=None*)
Bases: object

Base class for MySQL Connections subclasses.

Should not be used directly but overloaded, changing the open_connection part. Examples over subclasses are

MySQLTCPConnection MySQLUNIXConnection

close_connection()

do_handshake()

get_address()

open_connection()

recv()
Receive packets using the socket from the server.

send(*buf*)
Send packets using the socket to the server.

set_connection_timeout(*timeout*)

set_protocol(*prtcls*)

class mysql.connector.connection.**MySQLTCPConnection**(*prtcls=None*, *host='127.0.0.1'*, *port=3306*)
Bases: mysql.connector.connection.MySQLBaseConnection

Opens a TCP connection to the MySQL Server.

get_address()

open_connection()
Opens a TCP Connection and checks the MySQL handshake.

class mysql.connector.connection.**MySQLUnixConnection**(*prtcls=None*, *unix_socket='/tmp/mysql.sock'*)
Bases: mysql.connector.connection.MySQLBaseConnection

Opens a connection through the UNIX socket of the MySQL Server.

get_address()

open_connection()
Opens a UNIX socket and checks the MySQL handshake.

constants Module

Various MySQL constants and character sets

class mysql.connector.constants.**CharacterSet**
Bases: mysql.connector.constants._constants

List of supported character sets with their collations. This maps to the character set we get from the server within the handshake packet.

To update this list, use the following query:

SELECT ID,CHARACTER_SET_NAME, COLLATION_NAME FROM INFORMATION_SCHEMA.COLLATIONS ORDER BY ID

This list is hardcoded because we want to avoid doing each time the above query to get the name of the character set used.

classmethod get_charset_info(*name*, *collation=None*)
Returns information about the charset and optional collation.

classmethod get_desc(*setid*)
Returns info string about the charset for given MySQL ID.

classmethod get_info(*setid*)
Returns information about the charset for given MySQL ID.

classmethod get_supported()
Returns a list with names of all supproted character sets.

class mysql.connector.constants.**ClientFlag**
Bases: mysql.connector.constants._constantflags

Client Options as found in the MySQL sources mysql-src/include/mysql_com.h

COMPRESS = 32

CONNECT_WITH_DB = 8

FOUND_ROWS = 2

IGNORE_SIGPIPE = 4096

IGNORE_SPACE = 256

INTERACTIVE = 1024

LOCAL_FILES = 128

LONG_FLAG = 4

LONG_PASSWD = 1

MULTI_RESULTS = 131072

MULTI_STATEMENTS = 65536

NO_SCHEMA = 16

ODBC = 64

PROTOCOL_41 = 512

REMEMBER_OPTIONS = 2147483648

RESERVED = 16384

SECURE_CONNECTION = 32768

SSL = 2048

SSL_VERIFY_SERVER_CERT = 1073741824

TRANSACTIONS = 8192

default = [1, 4, 8, 512, 8192, 32768, 65536, 131072]

desc = {'NO_SCHEMA': (16, "Don't allow database.table.column"), 'IGNORE_SIGPIPE': (4096, 'IGNORE sigpipes'),

classmethod get_default()

class mysql.connector.constants.**FieldFlag**
Bases: mysql.connector.constants._constantflags

Field flags as found in MySQL sources mysql-src/include/mysql_com.h

AUTO_INCREMENT = 512

BINARY = 128

BINCMP = 131072

BLOB = 16

ENUM = 256

FIELD_IN_ADD_INDEX = 1048576

FIELD_IN_PART_FUNC = 524288

FIELD_IS_RENAMED = 2097152

GET_FIXED_FIELDS = 262144

GROUP = 16384

MULTIPLE_KEY = 8

NOT_NULL = 1

NO_DEFAULT_VALUE = 4096

NUM = 16384

ON_UPDATE_NOW = 8192

PART_KEY = 32768

PRI_KEY = 2

SET = 2048

TIMESTAMP = 1024

UNIQUE = 65536

UNIQUE_KEY = 4

UNSIGNED = 32

ZEROFILL = 64

desc = {'GROUP': (16384, 'Intern: Group field'), 'FIELD_IN_PART_FUNC': (524288, 'Field part of partition func'), 'P

class mysql.connector.constants.**FieldType**

Bases: mysql.connector.constants._constants

BIT = 16

BLOB = 252

DATE = 10

DATETIME = 12

DECIMAL = 0

DOUBLE = 5

ENUM = 247

FLOAT = 4

GEOMETRY = 255

INT24 = 9

LONG = 3

LONGLONG = 8

LONG_BLOB = 251

MEDIUM_BLOB = 250

NEWDATE = 14

NEWDECIMAL = 246

NULL = 6

SET = 248

SHORT = 2

STRING = 254

TIME = 11

TIMESTAMP = 7

TINY = 1

TINY_BLOB = 249

VARCHAR = 15

VAR_STRING = 253

YEAR = 13

desc = {'LONGLONG': (8, 'LONGLONG'), 'SHORT': (2, 'SHORT'), 'LONG': (3, 'LONG'), 'DATE': (10, 'DATE'), 'N

classmethod get_binary_types()

classmethod get_number_types()

classmethod get_string_types()

classmethod get_timestamp_types()

prefix = 'FIELD_TYPE_'

class mysql.connector.constants.**RefreshOption**

Bases: mysql.connector.constants._constants

Options used when sending the COM_REFRESH server command.

GRANT = 1

HOST = 8

LOG = 2

SLAVE = 64

STATUS = 16

TABLES = 4

THREADS = 32

desc = {'STATUS': (16, 'Flush status variables'), 'TABLES': (4, 'close all tables'), 'SLAVE': (64, 'Reset master info and

class mysql.connector.constants.**ServerCmd**

Bases: mysql.connector.constants._constants

BINLOG_DUMP = 18

CHANGE_USER = 17

CONNECT = 11

CONNECT_OUT = 20

CREATE_DB = 5

DAEMON = 29

DEBUG = 13

DELAYED_INSERT = 16

DROP_DB = 6

FIELD_LIST = 4

INIT_DB = 2

PING = 14

PROCESS_INFO = 10

PROCESS_KILL = 12

QUERY = 3

QUIT = 1

REFRESH = 7

REGISTER_SLAVE = 21

SET_OPTION = 27

SHUTDOWN = 8

SLEEP = 0

STATISTICS = 9

STMT_CLOSE = 25

STMT_EXECUTE = 23

STMT_FETCH = 28

STMT_PREPARE = 22

STMT_RESET = 26

STMT_SEND_LONG_DATA = 24

TABLE_DUMP = 19

TIME = 15

class mysql.connector.constants.**ServerFlag**

Bases: mysql.connector.constants._constantflags

Server flags as found in the MySQL sources mysql-src/include/mysql_com.h

MORE_RESULTS_EXISTS = 8

QUERY_NO_GOOD_INDEX_USED = 16

QUERY_NO_INDEX_USED = 32

STATUS_AUTOCOMMIT = 2

STATUS_CURSOR_EXISTS = 64

STATUS_DB_DROPPED = 256

STATUS_IN_TRANS = 1

STATUS_LAST_ROW_SENT = 128

STATUS_NO_BACKSLASH_ESCAPES = 512

desc = {'SERVER_STATUS_DB_DROPPED': (256, 'A database was dropped'), 'SERVER_MORE_RESULTS_EXISTS

conversion Module

Converting MySQL and Python types

class mysql.connector.conversion.**ConverterBase**(*charset='utf8'*, *use_unicode=True*)
Bases: object

escape(*buf*)

quote(*buf*)

set_charset(*charset*)

set_unicode(*value=True*)

to_mysql(*value*)

to_python(*vtype*, *value*)

class mysql.connector.conversion.**MySQLConverter**(*charset=None*, *use_unicode=True*)
Bases: mysql.connector.conversion.ConverterBase

A converted class grouping: o escape method: for escpaing values send to MySQL o quoting method: for quoting values send to MySQL in statements o conversion mapping: maps Python and MySQL data types to

function for converting them.

This class should be overloaded whenever one needs differences in how values are to be converted. Each MySQLConnection object has a default_converter property, which can be set like

MySQL.converter(CustomMySQLConverter)

escape(*value*)
Escapes special characters as they are expected to by when MySQL receives them. As found in MySQL source mysys/charset.c

Returns the value if not a string, or the escaped string.

quote(*buf*)

Quote the parameters for commands. General rules: o numbers are returns as str type (because operation expect it) o None is returned as str('NULL') o String are quoted with single quotes '<string>'

Returns a string.

to_mysql(*value*)

to_python(*flddsc*, *value*)
Converts a given value coming from MySQL to a certain type in Python. The flddsc contains additional information for the field in the table. It's an element from MySQLCursor.description.

Returns a mixed value.

cursor Module

Cursor classes

class mysql.connector.cursor.**CursorBase**
Bases: object

Base for defining MySQLCursor. This class is a skeleton and defines methods and members as required for the Python Database API Specification v2.0.

It's better to inherite from MySQLCursor.

callproc(*procname*, *args=()*)

close()

execute(*operation*, *params=()*)

executemany(*operation*, *seqparams*)

fetchall()

fetchmany(*size=1*)

fetchone()

nextset()

reset()

setinputsizes(*sizes*)

setoutputsize(*size*, *column=None*)

class mysql.connector.cursor.**MySQLCursor**(*db=None*)
Bases: mysql.connector.cursor.CursorBase

Default cursor which fetches all rows and stores it for later usage. It uses the converter set for the MySQLConnection to map MySQL types to Python types automatically.

This class should be inherited whenever other functionallity is required. An example would to change the fetch* member functions to return dictionaries instead of lists of values.

Implements the Python Database API Specification v2.0.

Possible parameters are:

db A MySQLConnection instance.

callproc(*procname*, *args=()*)
Calls a stored procedue with the given arguments

The arguments will be set during this session, meaning they will be called like _<procname>__arg<nr> where <nr> is an enumeration (+1) of the arguments.

Coding Example: 1) Definining the Stored Routine in MySQL: CREATE PROCEDURE multiply(IN pFac1 INT, IN pFac2 INT, OUT pProd INT) BEGIN

SET pProd := pFac1 * pFac2;

END

2) Executing in Python: args = (5,5,0) # 0 is to hold pprod cursor.callproc(multiply, args) print cursor.fetchone()

The last print should output ('5', '5', 25L)

Does not return a value, but a result set will be available when the CALL-statement execute succesfully. Raises exceptions when something is wrong.

close()
Close the cursor, disconnecting it from the MySQL object.

Returns True when succesful, otherwise False.

execute(*operation*, *params=None*)
Executes the given operation. The parameters given through params are used to substitute %%s in the operation string. For example, getting all rows where id is 5:

cursor.execute("SELECT * FROM t1 WHERE id = %s", (5,))

If warnings where generated, and db.get_warnings is True, then self._warnings will be a list containing these warnings.

Raises exceptions when any error happens.

executemany(*operation*, *seq_params*)
Loops over seq_params and calls excute()

fetchall()

fetchmany(*size=None*)

fetchone()

fetchwarnings()

getlastrowid()

next()
Used for iterating over the result set. Calles self.fetchone() to get the next row.

set_connection(*db*)

class mysql.connector.cursor.**MySQLCursorBuffered**(*db=None*)
Bases: mysql.connector.cursor.MySQLCursor

Cursor which fetches rows within execute()

fetchall()

fetchmany(*size=None*)

reset()

dbapi Module

DB API v2.0 required

mysql.connector.dbapi.**DateFromTicks**(*ticks*)

mysql.connector.dbapi.**TimeFromTicks**(*ticks*)

mysql.connector.dbapi.**TimestampFromTicks**(*ticks*)

errors Module

Python exceptions

class mysql.connector.errors.**ClientError**
Bases: object

client_errors = {2048: 'Invalid connection handle', 2049: "Connection using old (pre-4.1.1) authentication protocol r

classmethod get_error_msg(*errno*, *values=None*)

exception mysql.connector.errors.**DataError**(*m=None*, *errno=None*, *values=None*)
Bases: mysql.connector.errors.DatabaseError

exception mysql.connector.errors.**DatabaseError**(*m=None*, *errno=None*, *values=None*)
Bases: mysql.connector.errors.Error

exception mysql.connector.errors.**Error**(*m*, *errno=None*, *values=None*)
Bases: exceptions.StandardError

exception mysql.connector.errors.**IntegrityError**(*m=None*, *errno=None*, *values=None*)
Bases: mysql.connector.errors.DatabaseError

exception mysql.connector.errors.**InterfaceError**(*m=None*, *errno=None*, *values=None*)
Bases: mysql.connector.errors.Error

exception mysql.connector.errors.**InternalError**(*m=None*, *errno=None*, *values=None*)
Bases: mysql.connector.errors.DatabaseError

exception mysql.connector.errors.**NotSupportedError**(*m=None*, *errno=None*, *values=None*)
Bases: mysql.connector.errors.DatabaseError

exception mysql.connector.errors.**OperationalError**(*m=None*, *errno=None*, *values=None*)
Bases: mysql.connector.errors.DatabaseError

exception mysql.connector.errors.**ProgrammingError**(*m=None*, *errno=None*, *values=None*)
Bases: mysql.connector.errors.DatabaseError

exception mysql.connector.errors.**Warning**
Bases: exceptions.StandardError

mysql.connector.errors.**get_mysql_exception**(*errno*, *msg*)

mysql Module

Main classes for interacting with MySQL

class mysql.connector.mysql.**MySQL**(**args*, ***kwargs*)
Bases: mysql.connector.mysql.MySQLBase

Class implementing Python DB API v2.0.

close()

commit()
Shortcut for executing COMMIT.

connect(*dsn=''*, *user=''*, *password=''*, *host='127.0.0.1'*, *port=3306*, *db=None*, *database=None*, *use_unicode=True*, *charset='utf8'*, *get_warnings=False*, *raise_on_warnings=False*, *autocommit=False*, *unix_socket=None*, *connection_timeout=None*, *client_flags=0*, *buffered=False*)

Establishes a connection to the MySQL Server. Called also when instansiating a new MySQLConnection object through the __init__ method.

Possible parameters are:

dsn (not used)

user The username used to authenticate with the MySQL Server.

password The password to authenticate the user with the MySQL Server.

host The hostname or the IP address of the MySQL Server we are connecting with. (default 127.0.0.1)

port TCP port to use for connecting to the MySQL Server. (default 3306)

database db

Initial database to use once we are connected with the MySQL Server. The db argument is synonym, but database takes precedence.

use_unicode If set to true, string values received from MySQL will be returned as Unicode strings. Default: True

charset Which character shall we use for sending data to MySQL. One can still override this by using the SET NAMES command directly, but this is discouraged. Instead, use the set_charset() method if you want to change it. Default: Whatever the MySQL server has default.

get_warnings If set to true, whenever a query gives a warning, a SHOW WARNINGS will be done to fetch them. They will be available as MySQLCursor.warnings. The default is to ignore these warnings, for debugging it's good to enable it though, or use strict mode in MySQL to make most of these warnings errors. Default: False

raise_on_warnings If set to True, warnings will be raised as exceptions. raise_on_warings overrides get_warnings. Default: False

autocommit Auto commit is OFF by default, which is required by the Python Db API 2.0 specification. Default: False

unix_socket Full path to the MySQL Server UNIX socket. By default TCP connection will be used using the address specified by the host argument.

connection_timeout Timeout for the TCP and UNIX socket connection.

client_flags

Allows to set flags for the connection. Check following for possible flags:

```
>>> from mysql.connector.constants import ClientFlag
>>> print '
```

'.join(ClientFlag.get_full_info())

buffered When set to True .execute() will fetch the rows immediatly.

cursor()

remove_cursor(*c*)

rollback()
Shortcut for executing ROLLBACK

class mysql.connector.mysql.**MySQLBase**
Bases: object

MySQLBase

connect()
To be implemented while subclassing MySQLBase.

disconnect()
Disconnect from the MySQL server.

get_characterset_info()

get_server_info()
Returns the server version as a string

get_server_threadid()
Returns the MySQL threadid of the connection.

get_server_version()
Returns the server version as a tuple

is_connected()
Check whether we are connected to the MySQL server.

isset_client_flag(*flag*)

ping()
Check whether we are connected to the MySQL server.

set_autocommit(*switch*)
Set auto commit on or off. The argument 'switch' must be a boolean type.

set_buffered(*val=False*)
Sets whether cursor .execute() fetches rows

set_charset(*name*)
Set the character set used for the connection. This is the recommended way of change it per connection basis. It does execute SET NAMES internally, but it's good not to use this command directly, since we are setting some other members accordingly.

set_charset_info(*info=None*, *charset=None*)

set_client_flag(*flag*)

set_client_flags(*flags*)

set_connection_timeout(*timeout*)

set_converter_class(*convclass*)
Set the converter class to be used. This should be a class overloading methods and members of conversion.MySQLConverter.

set_database(*database*)
Set the database to be used after connection succeeded.

set_host(*host*)
Set the host for connection to the MySQL server.

set_login(*username=None*, *password=None*)
Set the username and/or password for the user connecting to the MySQL Server.

set_port(*port*)
Set the TCP port to be used when connecting to the server, usually 3306.

set_unicode(*value=True*)
Set whether we return string fields as unicode or not. Default is True.

set_unixsocket(*loc*)
Set the UNIX Socket location. Does not check if it exists.

set_warnings(*fetch=False*, *raise_on_warnings=False*)
Set how to handle warnings coming from MySQL

Set wheter we should get warnings whenever an operation produced some. If you set raise_on_warnings to True, any warning will be raised as a DataError exception.

unset_client_flag(*flag*)

protocol Module

Implementing the MySQL Client/Server protocol

class mysql.connector.protocol.**Auth**(*packet=None*, *client_flags=0*, *pktnr=0*)
Bases: mysql.connector.protocol.PacketOut

create(*username=None*, *password=None*, *database=None*, *seed=None*, *charset=33*)

scramble(*passwd*, *seed*)

set_client_flags(*flags*)

set_login(*username*, *password*, *database=None*)

class mysql.connector.protocol.**BasePacket**
Bases: object

is_valid(*buf=None*)

class mysql.connector.protocol.**ChangeUserPacket**
Bases: mysql.connector.protocol.Auth

create(*username=None*, *password=None*, *database=None*, *charset=8*, *seed=None*)

class mysql.connector.protocol.**CommandPacket**(*cmd=None*, *arg=None*)
Bases: mysql.connector.protocol.PacketOut

create()

set_argument(*arg*)

set_command(*cmd*)

class mysql.connector.protocol.**EOFPacket**(*buf=None*)
Bases: mysql.connector.protocol.PacketIn

class mysql.connector.protocol.**ErrorResultPacket**(*buf=None*)
Bases: mysql.connector.protocol.PacketIn

class mysql.connector.protocol.**FieldPacket**(*buf=None*)
Bases: mysql.connector.protocol.PacketIn

get_description()
Returns a description as a list useful for cursors.

This function returns a list as defined in the Python Db API v2.0 specification.

class mysql.connector.protocol.**Handshake**(*buf=None*)
Bases: mysql.connector.protocol.PacketIn

get_dict()

class mysql.connector.protocol.**KillPacket**(*arg*)
Bases: mysql.connector.protocol.CommandPacket

create()

set_argument(*arg*)

class mysql.connector.protocol.**MySQLProtocol**(*conn*, *handshake=None*)
Bases: object

Class handling the MySQL Protocol.

MySQL v4.1 Client/Server Protocol is currently supported.

cmd_change_user(*username*, *password*, *database=None*)
Change the user with given username and password to another optional database.

cmd_debug()
Send DEBUG command to the MySQL Server

Needs SUPER privileges. The output will go to the MySQL server error log.

Returns True when it was succesful.

cmd_init_db(*database*)
Send command to server to change databases.

cmd_ping()
Ping the MySQL server to check if the connection is still alive.

Returns True when alive, False when server doesn't respond.

cmd_process_info()
Gets the process list from the MySQL Server.

Returns a list of dictionaries which corresponds to the output of SHOW PROCESSLIST of MySQL. The data is converted to Python types.

cmd_process_kill(*mypid*)
Kills a MySQL process using it's ID.

The mypid must be an integer.

cmd_query(*query*)
Sends a query to the server.

Returns a tuple, when the query returns a result. The tuple consist number of fields and a list containing their descriptions. If the query doesn't return a result set, the an OKResultPacket will be returned.

cmd_quit()
Closes the current connection with the server.

cmd_refresh(*opts*)
Send the Refresh command to the MySQL server.

The argument should be a bitwise value using the protocol.RefreshOption constants.

Usage:

RefreshOption = mysql.connector.RefreshOption refresh = RefreshOption.LOG | RefreshOption.THREADS db.cmd_refresh(refresh)

cmd_shutdown()
Shuts down the MySQL Server.

Careful with this command if you have SUPER privileges! (Which your scripts probably don't need!)

Returns True if it succeeds.

cmd_statistics()
Sends statistics command to the MySQL Server

Returns a dictionary with various statistical information.

do_auth(*username=None*, *password=None*, *database=None*, *client_flags=None*, *charset=33*)
Make and send the authentication using information found in the handshake packet.

handle_handshake(*buf*)
Check whether the buffer is a valid handshake. If it is, we set some member variables for later usage. The handshake packet is returned for later usuage, e.g. authentication.

handle_header(*buf*)
Takes a buffer and readers information from header.

Returns a tuple (pktsize, pktnr)

is_eof(*buf*)
Check if the given buffer is a MySQL EOF Packet. It should start with þ and be smaller 9 bytes.

Returns boolean.

is_error(*buf*)
Check if the given buffer is a MySQL Error Packet.

Buffer should start with ÿ.

Returns boolean.

is_ok(*buf*)
Check if the given buffer is a MySQL OK Packet. It should start with .

Returns boolean.

result_get_row()
Get data for 1 row

Get one row's data. Should be called after getting the field descriptions.

Returns a tuple with 2 elements: a row's data and the EOF packet.

result_get_rows(*cnt=None*)
Get all rows

Returns a tuple with 2 elements: a list with all rows and the EOF packet.

set_handshake(*handshake*)
Gather data from the given handshake.

class mysql.connector.protocol.**OKResultPacket**(*buf=None*)
Bases: mysql.connector.protocol.PacketIn

class mysql.connector.protocol.**PacketIn**(*buf=None*, *pktnr=0*)
Bases: mysql.connector.protocol.BasePacket

class mysql.connector.protocol.**PacketOut**(*buf=None*, *pktnr=0*)
Bases: mysql.connector.protocol.BasePacket

Each packet type used in the MySQL Client Protocol is build on the Packet class. It defines lots of useful functions for parsing and sending data to and from the MySQL Server.

add(*s*)

add_1_int(*i*)

add_2_int(*i*)

add_3_int(*i*)

add_4_int(*i*)

add_null(*nr=1*)

get()

get_header()

set(*buf*)

utils Module

Utilities

mysql.connector.utils.**int1read**(*c*)
Takes a bytes and returns it was an integer.

Returns integer.

mysql.connector.utils.**int1store**(*i*)
Takes an unsigned byte (1 byte) and packs it as string.

Returns string.

mysql.connector.utils.**int2read**(*s*)
Takes a string of 2 bytes and unpacks it as unsigned integer.

Returns integer.

mysql.connector.utils.**int2store**(*i*)
Takes an unsigned short (2 bytes) and packs it as string.

Returns string.

mysql.connector.utils.**int3read**(*s*)
Takes a string of 3 bytes and unpacks it as integer.

Returns integer.

mysql.connector.utils.**int3store**(*i*)
Takes an unsigned integer (3 bytes) and packs it as string.

Returns string.

mysql.connector.utils.**int4read**(*s*)
Takes a string of 4 bytes and unpacks it as integer.

Returns integer.

mysql.connector.utils.**int4store**(*i*)
Takes an unsigned integer (4 bytes) and packs it as string.

Returns string.

mysql.connector.utils.**int8read**(*s*)
Takes a string of 8 bytes and unpacks it as integer.

Returns integer.

mysql.connector.utils.**intread**(*s*)
Takes a string and unpacks it as an integer.

This function uses int1read, int2read, int3read and int4read by checking the length of the given string.

Returns integer.

mysql.connector.utils.**intstore**(*i*)
Takes an unsigned integers and packs it as a string.

This function uses int1store, int2store, int3store and int4store depending on the integer value.

returns string.

mysql.connector.utils.**read_bytes**(*buf*, *size*)
Reads bytes from a buffer.

Returns a tuple with buffer less the read bytes, and the bytes.

mysql.connector.utils.**read_int**(*buf*, *size*)
Take a buffer and reads an integer of a certain size (1 <= size <= 4).

Returns a tuple (truncated buffer, int)

mysql.connector.utils.**read_lc_int**(*buf*)
Takes a buffer and reads an length code string from the start.

Returns a tuple with buffer less the integer and the integer read.

mysql.connector.utils.**read_lc_string**(*buf*)
Takes a buffer and reads a length coded string from the start.

This is how Length coded strings work

If the string is 250 bytes long or smaller, then it looks like this:

<– 1b –> +———-+———————- | length | a string goes here +———-+———————-

If the string is bigger than 250, then it looks like this:

<- 1b -><- 2/3/4 -> +——+———-+———————- | type | length | a string goes here +——+———-+———————-

if type == ü: length is code in next 2 bytes

elif type == ý: length is code in next 3 bytes

elif type == þ: length is code in next 4 bytes

NULL has a special value. If the buffer starts with û then it's a NULL and we return None as value.

Returns a tuple (trucated buffer, string).

mysql.connector.utils.**read_lc_string_list**(*buf*)
Reads all length encoded strings from the given buffer.

This is exact same function as read_lc_string() but duplicated in hopes for performance gain when reading results.

mysql.connector.utils.**read_string**(*buf*, *end=None*, *size=None*)
Reads a string up until a character or for a given size.

Returns a tuple (trucated buffer, string).

CHAPTER

SIX

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX