

## NAME

SDFFileUtil

## SYNOPSIS

```
use SDFFileUtil ;

use SDFFileUtil qw(:all);
```

## DESCRIPTION

SDFFileUtil module provides the following functions:

GenerateCmpdAtomLine, GenerateCmpdBondLine, GenerateCmpdChargePropertyLines, GenerateCmpdCommentsLine, GenerateCmpdCountsLine, GenerateCmpdDataHeaderLabelsAndValuesLines, GenerateCmpdIsotopePropertyLines, GenerateCmpdMiscInfoLine, GenerateCmpdMolNameLine, GenerateCmpdRadicalPropertyLines, GenerateEmptyCtabBlockLines, GenerateMiscLineDateStamp, GetAllAndCommonCmpdDataHeaderLabels, GetCmpdDataHeaderLabels, GetCmpdDataHeaderLabelsAndValues, GetCmpdFragments, GetCtabLinesCount, GetUnknownAtoms, InternalBondOrderToMDLBondType, InternalBondTypeToMDLBondStereo, InternalChargeToMDLCharge, InternalSpinMultiplicityToMDLRadical, MDLBondStereoToInternalBondType, MDLBondTypeToInternalBondOrder, MDLChargeToInternalCharge, MDLRadicalToInternalSpinMultiplicity, ParseCmpdAtomLine, ParseCmpdBondLine, ParseCmpdChargePropertyLine, ParseCmpdCommentsLine, ParseCmpdCountsLine, ParseCmpdIsotopePropertyLine, ParseCmpdMiscInfoLine, ParseCmpdMolNameLine, ParseCmpdRadicalPropertyLine, ReadCmpdString, WashCmpd

## METHODS

## GenerateCmpdAtomLine

```
$Line = GenerateCmpdAtomLine($AtomSymbol, $AtomX, $AtomY,
                             $AtomZ, [$MassDifference, $Charge, $StereoParity]);
```

Returns a formatted atom data line containing all the input values

## GenerateCmpdBondLine

```
$Line = GenerateCmpdBondLine($FirstAtomNum, $SecondAtomNum,
                             $BondType, [$BondStereo]);
```

Returns a formatted bond data line containing all the input values

## GenerateCmpdChargePropertyLines

```
@Lines = GenerateCmpdChargePropertyLines($ChargeValuePairsRef);
```

Returns a formatted M CHG property lines corresponding to successive pairs of atom number and charge values specified by a reference to an array

## GenerateCmpdCommentsLine

```
$Line = GenerateCmpdCommentsLine($Comments);
```

Returns a formatted comments data line

## GenerateCmpdCountsLine

```
$Line = GenerateCmpdCountsLine($AtomCount, $BondCount,
                              $ChiralFlag, [$PropertyCount, $Version]);
```

Returns a formatted line containing all the input values. The default values of 999 and V2000 are used for *PropertyCount* and *Version*.

## GenerateCmpdDataHeaderLabelsAndValuesLines

```
@Lines = GenerateCmpdDataHeaderLabelsAndValuesLines(
    $DataHeaderLabelsRef, $DataHeaderLabelsAndValuesRef,
    [$SortDataLabels]);
```

Returns formatted data lines containing header label and values lines corresponding to all data header labels in array reference *DataHeaderLabelsRef* with values in hash reference *DataHeaderLabelsAndValuesRef*. By default, data header labels are not sorted and correspond to the label order in array reference *DataHeaderLabelsRef*.

## GenerateCmpdIsotopePropertyLines

```
@Lines = GenerateCmpdIsotopePropertyLines($IsotopeValuePairsRef);
```

Returns a formatted M ISO property lines corresponding to successive pairs of atom number and isotope values specified by a reference to an array

## GenerateCmpdMiscInfoLine

```
$Line = GenerateCmpdMiscInfoLine([$ProgramName, $UserInitial,
```

```
$Code]);
```

Returns a formatted line containing specified user initial, program name, date and code. Default values are: *ProgramName - MayaChem; UserInitial - NULL; Code - 2D*.

#### GenerateCmpdMolNameLine

```
$Line = GenerateCmpdMolNameLine($MolName);
```

Returns a formatted molecule name data line

#### GenerateCmpdRadicalPropertyLines

```
@Lines = GenerateCmpdRadicalPropertyLines($RadicalValuePairsRef);
```

Returns a formatted M CHG property lines corresponding to successive pairs of atom number and multiplicity values specified by a reference to an array

#### GenerateEmptyCtabBlockLines

```
$Lines = GenerateCmpdMiscInfoLine([$Date]);
```

Returns formatted lines representing empty CTAB block

#### GenerateMiscLineDateStamp

```
$Line = GenerateMiscLineDateStamp();
```

Returns date stamp for misc line.

#### GetAllAndCommonCmpdDataHeaderLabels

```
($CmpdCount, $DataFieldLabelsArrayRef,
 $CommonDataFieldLabelsArrayRef) =
  GetAllAndCommonCmpdDataHeaderLabels(\*SDFILE);
```

Returns number of compounds, a reference to an array containing all unique data header label and a reference to an array containing common data field labels for all compounds in SD file.

#### GetCmpdDataHeaderLabels

```
(@Labels) = GetCmpdDataHeaderLabels(\@CmpdLines);
```

Returns an array containing data header labels for a compound

#### GetCmpdDataHeaderLabelsAndValues

```
(%DataValues) = GetCmpdDataHeaderLabelsAndValues(\@CmpdLines);
```

Returns a hash containing data header labels and values for a compound

#### GetCmpdFragments

```
($FragmentCount, $FragmentString) = GetCmpdFragments(\@CmpdLines);
```

Figures out the number of disconnected fragments and return their values along with the atom numbers in a string delimited by new line character. Fragment data in FragmentString is sorted on based on its size.

#### GetCtabLinesCount

```
$CtabLinesCount = GetCtabLinesCount(\@CmpdLines);
```

Returns number of lines present between the 4th line and the line containing "M END"

#### GetUnknownAtoms

```
($UnknownAtomCount, $UnknownAtoms, $UnknownAtomLines) =
  GetUnknownAtoms(\@CmpdLines);
```

Returns a list of values containing information about atoms which contain special element symbols not present in the periodic table

#### InternalBondOrderToMDLBondType

```
$MDLBondType = InternalBondOrderToMDLBondType($InternalBondOrder);
```

Returns value of *MDLBondType* corresponding to *InternalBondOrder*

InternalBondOrder	MDLBondType
1	1
2	2

3	3
1.5	4

**InternalBondTypeToMDLBondStereo**

```
$MDLBondStereo = InternalBondTypeToMDLBondStereo($InternalBondType);
```

Returns value of *MDLBondStereo* corresponding to *InternalBondType* using following mapping:

InternalBondType	MDLBondStereo
SingleUp	1
SingleWavy	4
SingleDown	6
DoubleCross	3

**InternalChargeToMDLCharge**

```
$MDLCharge = InternalChargeToMDLCharge($InternalCharge);
```

Returns value of *MDLCharge* corresponding to *InternalCharge* using following mapping:

InternalCharge	MDLCharge
3	1
2	2
1	3
-1	5
-2	6
-3	7

**InternalSpinMultiplicityToMDLRadical**

```
$MDLRadical = InternalSpinMultiplicityToMDLRadical(
    $InternalSpinMultiplicity);
```

Returns value of *MDLRadical* corresponding to *InternalSpinMultiplicity*. These value are equivalent.

**MDLBondStereoToInternalBondType**

```
$InternalBondType = MDLBondStereoToInternalBondType($MDLBondStereo);
```

Returns value of *InternalBondType* corresponding to *MDLBondStereo* using mapping shown for *InternalBondTypeToMDLBondStereo* function

**MDLBondTypeToInternalBondOrder**

```
$InternalBondOrder = MDLBondTypeToInternalBondOrder($MDLBondType);
```

Returns value of *InternalBondOrder* corresponding to *MDLBondType* using mapping shown for *InternalBondOrderToMDLBondType* function

**MDLChargeToInternalCharge**

```
$InternalCharge = MDLChargeToInternalCharge($MDLCharge);
```

Returns value of *InternalCharge* corresponding to *MDLCharge* using mapping shown for *InternalChargeToMDLCharge* function

**MDLRadicalToInternalSpinMultiplicity**

```
$InternalSpinMultiplicity = MDLRadicalToInternalSpinMultiplicity(
    $MDLRadical);
```

Returns value of *InternalSpinMultiplicity* corresponding to *MDLRadical*. These value are equivalent.

**ParseCmpdAtomLine**

```
($AtomSymbol, $AtomX, $AtomY, $AtomZ, $MassDifference, $Charge,
    $StereoParity) = ParseCmpdAtomLine($AtomDataLine);
```

Parses compound data line containing atom information and returns a list of values

**ParseCmpdBondLine**

```
($FirstAtomNum, $SecondAtomNum, $BondType) =
    ParseCmpdBondLine($BondDataLine);
```

Parses compound data line containing bond information and returns a list of values

### ParseCmpdCommentsLine

```
$Comments = ParseCmpdCommentsLine($CommentsDataLine);
```

Returns the comment string

### ParseCmpdChargePropertyLine

```
@AtomNumAndValuePairs = ParseCmpdChargePropertyLine($ChargeDataLine);
```

Parses charge property line in CTAB generic properties block and returns an array with successive pairs of values corresponding to atom number and its charge

### ParseCmpdCountsLine

```
($AtomCount, $BondCount, $ChiralFlag, $PropertyCount, $Version) =  
ParseCmpdCountsLine(\@CountDataLines);
```

Returns a list of values containing count information

### ParseCmpdMiscInfoLine

```
($UserInitial, $ProgramName, $Date, $Code, $ScalingFactor1, $ScalingFactor2,  
$Energy, $RegistryNum) = ParseCmpdMiscInfoLine($Line);
```

Returns a list of values containing miscellaneous information.

### ParseCmpdIsotopePropertyLine

```
@AtomNumAndValuePairs = ParseCmpdIsotopePropertyLine($IsotopeDataLine);
```

Parses isotopic property line in CTAB generic properties block and returns an array with successive pairs of values corresponding to atom number and absolute mass of atom isotope

### ParseCmpdMolNameLine

```
$MolName = ParseCmpdMolNameLine($Line);
```

Returns a string containing molecule name

### ParseCmpdRadicalPropertyLine

```
@AtomNumAndValuePairs = ParseCmpdRadicalPropertyLine($RadicalDataLine);
```

Parses radical property line in CTAB generic properties block and returns an array with successive pairs of values corresponding to atom number and radical number value

### ReadCmpdString

```
$CmpdString = ReadCmpdString(\*SDFILEHANDLE);
```

Returns a string containing all the data lines for the next available compound in an already open file indicated by SDFILEHANDLE. A NULL string is returned on EOF.

### WashCmpd

```
($FragmentCount, $Fragments, $WashedCmpdString) =  
WashCmpd(\@CmpdLines);
```

Figures out the number of disconnected fragments and return their values along with the atom numbers in a string delimited by new line character. Fragment data in FragmentString is sorted on based on its size.

## AUTHOR

Manish Sud <msud@san.rr.com>

## SEE ALSO

TextUtil.pm

## COPYRIGHT

Copyright (C) 2004-2008 Manish Sud. All rights reserved.

This file is part of MayaChemTools.

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