

NAME

FingerprintsFPFileIO

SYNOPSIS

```
use FileIO::FingerprintsFPFileIO;

use FileIO::FingerprintsFPFileIO qw(:all);
```

DESCRIPTION

FingerprintsFPFileIO class provides the following methods:

new, GetFingerprints, GetFingerprintsString, GetHeaderDataKeyValue, GetHeaderDataKeys, GetHeaderDataKeysAndValues, GetPartialFingerprintsString, GetRequiredHeaderDataKeys, GetRequiredHeaderDataKeysAndValues, IsFingerprintsDataValid, IsFingerprintsFPFile, IsFingerprintsFileDataValid, IsHeaderDataKeyPresent, Next, Read, SetBitStringFormat, SetBitsOrder, SetCompoundID, SetDetailLevel, SetFingerprints, SetFingerprintsString, SetFingerprintsStringMode, SetPartialFingerprintsString, SetVectorStringFormat, WriteFingerprints, WriteFingerprintsString

The following methods can also be used as functions:

IsFingerprintsFPFile

FingerprintsFPFileIO class is derived from *FileIO* class and uses its methods to support generic file related functionality.

The MayaChemTools fingerprints file (FP) format with .fpf or .fp file extensions supports two types of fingerprints data: fingerprints bit-vectors and fingerprints vectors.

Example of FP file format containing fingerprints bit-vector string data:

```
#
# Package = MayaChemTools 7.4
# ReleaseDate = Oct 21, 2010
#
# TimeStamp = Mon Mar 7 15:14:01 2011
#
# FingerprintsStringType = FingerprintsBitVector
#
# Description = PathLengthBits:AtomicInvariantsAtomTypes:MinLength1:...
# Size = 1024
# BitStringFormat = HexadecimalString
# BitsOrder = Ascending
#
Cmpd1 9c8460989ec8a49913991a6603130b0a19e8051c89184414953800cc21510...
Cmpd2 000000249400840040100042011001001980410c000000001010088001120...
... ..
... ..
```

Example of FP file format containing fingerprints vector string data:

```
#
# Package = MayaChemTools 7.4
# ReleaseDate = Oct 21, 2010
#
# TimeStamp = Mon Mar 7 15:14:01 2011
#
# FingerprintsStringType = FingerprintsVector
#
# Description = PathLengthBits:AtomicInvariantsAtomTypes:MinLength1:...
# VectorStringFormat = IDsAndValuesString
# VectorValuesType = NumericalValues
#
Cmpd1 338;C F N O C:C C:N C=O CC CF CN CO C:C:C C:C:N C:CC C:CF C:CN C:
N:C C:NC CC:N CC=O CCC CCN CCO CNC NC=O O=CO C:C:C:C C:C:C:N C:C:CC...;
```

```

33 1 2 5 21 2 2 12 1 3 3 20 2 10 2 2 1 2 2 2 8 2 5 1 1 1 19 2 8 2 2 2 2
6 2 2 2 2 2 2 2 2 3 2 2 1 4 1 5 1 1 18 6 2 2 1 2 10 2 1 2 1 2 2 2 2 ...
Cmpd2 103;C N O C=N C=O CC CN CO CC=O CCC CCN CCO CNC N=CN NC=O NCN O=C
O C CC=O CCCC CCCN CCCO CCNC CNC=N CNC=O CNCN CCCC=O CCCCC CCCCN CC...;
15 4 4 1 2 13 5 2 2 15 5 3 2 2 1 1 1 2 17 7 6 5 1 1 1 2 15 8 5 7 2 2 2 2
1 2 1 1 3 15 7 6 8 3 4 4 3 2 2 1 2 3 14 2 4 7 4 4 4 4 1 1 1 2 1 1 1 ...
... ..
... ..

```

FP file data format consists of two main sections: header section and fingerprints string data section. The header section lines start with # and the first line not starting with # represents the start of fingerprints string data section. The header section contains both the required and optional information which is specified as key = value pairs. The required information describes fingerprints bit-vector and vector strings and used to generate fingerprints objects; the optional information is ignored during generation of fingerprints objects.

The key = value data specification in the header section and its processing follows these rules:

- o Leading and trailing spaces for key = value pairs are ignored
- o Key and value strings may contain spaces
- o Multiple key = value pairs on a single are delimited by semicolon

The default optional header data section key = value pairs are:

```

# Package = MayaChemTools 7.4
# ReleaseDate = Oct 21, 2010

```

The FingerprintsStringType key is required data header key for both fingerprints bit-vector and vector strings. Possible key values: *FingerprintsBitVector* or *FingerprintsVector*. For example:

```

# FingerprintsStringType = FingerprintsBitVector

```

The required data header keys for fingerprints bit-vector string are: Description, Size, BitStringFormat, and BitsOrder. Possible values for BitStringFormat: *HexadecimalString* or *BinaryString*. Possible values for BitsOrder: *Ascending* or *Descending*. The Description key contains information about various parameters used to generate fingerprints bit-vector string. The Size corresponds to number of fingerprints bits and is always less than or equal to number of bits in bit-vector string which might contain extra bits at the end to round off the size to make it multiple of 8. For example:

```

# Description = PathLengthBits:AtomicInvariantsAtomTypes:MinLength1:...
# Size = 1024
# BitStringFormat = HexadecimalString
# BitsOrder = Ascending

```

The required data header keys for fingerprints vector string are: Description, VectorStringFormat, and VectorValueType. Possible values for VectorStringFormat: *DsAndValuesString*, *IDsAndValuesPairsString*, *ValuesAndIDsString*, *ValuesAndIDsPairsString* or *ValuesString*. Possible values for VectorValueType: *NumericalValues*, *OrderedNumericalValues* or *AlphaNumericalValues*. The Description key contains information various parameters used to generate fingerprints vector string. For example:

```

# Description = PathLengthBits:AtomicInvariantsAtomTypes:MinLength1:...
# VectorStringFormat = IDsAndValuesString
# VectorValueType = NumericalValues

```

The fingerprints data section for fingerprints bit-vector string contains data in the following format:

```

... ..
CmpdID FingerprintsPartialBitVectorString
... ..

```

For example:

```

... ..

```

```
Cmpd1 9c8460989ec8a49913991a6603130b0a19e8051c89184414953800cc21510...
... ..
```

The fingerprints data section for fingerprints vector string contains data in the following format:

```
... ..
CmpdID Size;FingerprintsPartialVectorString
... ..
```

For example:

```
... ..
Cmpd1 338;C F N O C:C C:N C=O CC CF CN CO C:C:C C:C:N C:CC C:CF C:CN C:
N:C C:NC CC:N CC=O CCC CCN CCO CNC NC=O O=CO C:C:C:C C:C:C:N C:C:CC...;
33 1 2 5 21 2 2 12 1 3 3 20 2 10 2 2 1 2 2 2 8 2 5 1 1 1 19 2 8 2 2 2 2
6 2 2 2 2 2 2 2 3 2 2 1 4 1 5 1 1 18 6 2 2 1 2 10 2 1 2 1 2 2 2 2 ...
... ..
```

Unlike fingerprints bit-vector string, *Size* is specified for each partial fingerprints vector string: It may change from molecule to molecule for same type of fingerprints.

Values IDs are optional for fingerprints vector string containing *OrderedNumericalValues* or *AlphaNumericalValues*; however, they must be present for for *NumericalValues*. Due to various possible values for *VectorStringFormat*, the fingerprints data section for fingerprints vector string supports following type of data formats:

```
CmpdID Size;ID1 ID2 ID3...;Value1 Value2 Value3...
CmpdID Size;ID1 Value1 ID2 Value2 ID3 Value3... ..
CmpdID Size;ValuesAndIDsString: Value1 Value2 Value3...;ID1 ID2 ID3...
CmpdID Size;ValuesAndIDSPairsString: Value1 ID1 Value2 ID2 Value3 ID3... ..
CmpdID Size;Value1 Value2 Value3 ...
```

However, all the fingerprints vector string data present in FP file must correspond to only one of the formats shown above; multiple data formats in the same file are not allowed.

The current release of MayaChemTools supports the following types of fingerprint bit-vector and vector strings:

```
FingerprintsVector;AtomNeighborhoods:AtomicInvariantsAtomTypes:MinRadi
us0;MaxRadius2;41;AlphaNumericalValues;ValuesString;NR0-C.X1.BO1.H3-AT
C1:NR1-C.X3.BO3.H1-ATC1:NR2-C.X1.BO1.H3-ATC1:NR2-C.X3.BO4-ATC1 NR0-C.X
1.BO1.H3-ATC1:NR1-C.X3.BO3.H1-ATC1:NR2-C.X1.BO1.H3-ATC1:NR2-C.X3.BO4-A
TC1 NR0-C.X2.BO2.H2-ATC1:NR1-C.X2.BO2.H2-ATC1:NR1-C.X3.BO3.H1-ATC1:NR2
-C.X2.BO2.H2-ATC1:NR2-N.X3.BO3-ATC1:NR2-O.X1.BO1.H1-ATC1 NR0-C.X2.B...
```

```
FingerprintsVector;AtomTypesCount:AtomicInvariantsAtomTypes:ArbitraryS
ize;10;NumericalValues;IDsAndValuesString;C.X1.BO1.H3 C.X2.BO2.H2 C.X2
.BO3.H1 C.X3.BO3.H1 C.X3.BO4 F.X1.BO1 N.X2.BO2.H1 N.X3.BO3 O.X1.BO1.H1
O.X1.BO2;2 4 14 3 10 1 1 1 3 2
```

```
FingerprintsVector;AtomTypesCount:SLogPAtomTypes:ArbitrarySize;16;Nume
ricalValues;IDsAndValuesString;C1 C10 C11 C14 C18 C20 C21 C22 C5 CS F
N11 N4 O10 O2 O9;5 1 1 1 14 4 2 1 2 2 1 1 1 1 3 1
```

```
FingerprintsVector;AtomTypesCount:SLogPAtomTypes:FixedSize;67;OrderedN
umericalValues;IDsAndValuesString;C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11 C
12 C13 C14 C15 C16 C17 C18 C19 C20 C21 C22 C23 C24 C25 C26 C27 CS N1 N
2 N3 N4 N5 N6 N7 N8 N9 N10 N11 N12 N13 N14 NS O1 O2 O3 O4 O5 O6 O7 O8
O9 O10 O11 O12 OS F Cl Br I Hal P S1 S2 S3 Me1 Me2;5 0 0 0 2 0 0 0 0 1
1 0 0 1 0 0 0 14 0 4 2 1 0 0 0 0 2 0 0 0 1 0 0 0 0 0 1 0 0 0 0...
```

```
FingerprintsVector;EStateIndicies:ArbitrarySize;11;NumericalValues;IDs
AndValuesString;SaasCH SaasC SaasN SdO SdssC SsCH3 SsF SsOH SssCH2 SssN
H SsssCH;24.778 4.387 1.993 25.023 -1.435 3.975 14.006 29.759 -0.073 3
```


3 5 2 2 0 5 3 5 1 1 2 5 1 2 1 2 4 8 3 5 5 2 2 0 3 5 4 1

FingerprintsVector;MACCSKeyCount;322;OrderedNumericalValues;ValuesString;14 8 2 0 2 0 4 4 2 1 4 0 0 2 5 10 5 2 1 0 0 2 0 5 13 3 28 5 5 3 0 0
0 4 2 1 1 0 1 1 0 0 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 22 5 3 0 0 0 1 0
0
0
0 ...

FingerprintsBitVector;PathLengthBits:AtomicInvariantsAtomTypes:MinLength1:MaxLength8;1024;BinaryString;Ascending;001000010011010101011000110
0100010101011000101001011100110001000010001001101000001001001001001000
0010110100000111001001000001001010100100100000000011000000101001011100
0010000001000101010100000100111100110111011011011000000010110111001101
010110001100000001000100001100001010001110110000100000100010000000...

FingerprintsVector;PathLengthCount:AtomicInvariantsAtomTypes:MinLength1:MaxLength8;432;NumericalValues;IDsAndValuesPairsString;C.X1.BO1.H3 2
C.X2.BO2.H2 4 C.X2.BO3.H1 14 C.X3.BO3.H1 3 C.X3.BO4 10 F.X1.BO1 1 N.X
2.BO2.H1 1 N.X3.BO3 1 O.X1.BO1.H1 3 O.X1.BO2 2 C.X1.BO1.H3C.X3.BO3.H1
2 C.X2.BO2.H2C.X2.BO2.H2 1 C.X2.BO2.H2C.X3.BO3.H1 4 C.X2.BO2.H2C.X3.BO
4 1 C.X2.BO2.H2N.X3.BO3 1 C.X2.BO3.H1:C.X2.BO3.H1 10 C.X2.BO3.H1:C....

FingerprintsVector;PathLengthCount:MMFF94AtomTypes:MinLength1:MaxLength8;463;NumericalValues;IDsAndValuesPairsString;C5A 2 C5B 2 C=ON 1 CB 1
8 COO 1 CR 9 F 1 N5 1 NC=O 1 O=CN 1 O=CO 1 OC=O 1 OR 2 C5A:C5B 2 C5A:N
5 2 C5ACB 1 C5ACR 1 C5B:C5B 1 C5BC=ON 1 C5BCB 1 C=ON=O=CN 1 C=ONNC=O 1
CB:CB 18 CBF 1 CBNC=O 1 COO=O=CO 1 COOCR 1 COOOC=O 1 CRCR 7 CRN5 1 CR
OR 2 C5A:C5B:C5B 2 C5A:C5BC=ON 1 C5A:C5BCB 1 C5A:N5:C5A 1 C5A:N5CR ...

FingerprintsVector;TopologicalAtomPairs:AtomicInvariantsAtomTypes:MinDistance1:MaxDistance10;223;NumericalValues;IDsAndValuesString;C.X1.BO1
.H3-D1-C.X3.BO3.H1 C.X2.BO2.H2-D1-C.X2.BO2.H2 C.X2.BO2.H2-D1-C.X3.BO3.
H1 C.X2.BO2.H2-D1-C.X3.BO4 C.X2.BO2.H2-D1-N.X3.BO3 C.X2.BO3.H1-D1-...;
2 1 4 1 1 10 8 1 2 6 1 2 2 1 2 2 1 2 2 1 5 1 10 12 2 2 1 2 1 9 1 3 1
1 1 2 2 1 3 6 1 6 14 2 2 2 3 1 3 1 8 2 2 1 3 2 6 1 2 2 5 1 3 1 23 1...

FingerprintsVector;TopologicalAtomPairs:FunctionalClassAtomTypes:MinDistance1:MaxDistance10;144;NumericalValues;IDsAndValuesString;Ar-D1-Ar
Ar-D1-Ar.HBA Ar-D1-HBD Ar-D1-Hal Ar-D1-None Ar.HBA-D1-None HBA-D1-NI H
BA-D1-None HBA.HBD-D1-NI HBA.HBD-D1-None HBD-D1-None NI-D1-None No...;
23 2 1 1 2 1 1 1 1 2 1 1 7 28 3 1 3 2 8 2 1 1 1 5 1 5 24 3 3 4 2 13 4
1 1 4 1 5 22 4 4 3 1 19 1 1 1 1 2 2 3 1 1 8 25 4 5 2 3 1 26 1 4 1 ...

FingerprintsVector;TopologicalAtomTorsions:AtomicInvariantsAtomTypes;33;NumericalValues;IDsAndValuesString;C.X1.BO1.H3-C.X3.BO3.H1-C.X3.BO4-
C.X3.BO4 C.X1.BO1.H3-C.X3.BO3.H1-C.X3.BO4-N.X3.BO3 C.X2.BO2.H2-C.X2.BO
2.H2-C.X3.BO3.H1-C.X2.BO2.H2 C.X2.BO2.H2-C.X2.BO2.H2-C.X3.BO3.H1-O...;
2 2 1 1 2 2 1 1 3 4 4 8 4 2 2 6 2 2 1 2 1 1 2 1 1 2 6 2 4 2 1 3 1

FingerprintsVector;TopologicalAtomTorsions:ESStateAtomTypes;36;NumericalValues;IDsAndValuesString;aaCH-aaCH-aaCH-aaCH aaCH-aaCH-aaCH-aasC aaC
H-aaCH-aasC-aaCH aaCH-aaCH-aasC-aasC aaCH-aaCH-aasC-sF aaCH-aaCH-aasC-
ssNH aaCH-aasC-aasC-aasC aaCH-aasC-aasC-aasN aaCH-aasC-ssNH-dssC a...;
4 4 8 4 2 2 6 2 2 2 4 3 2 1 3 3 2 2 2 1 2 1 1 1 2 1 1 1 1 1 1 1 2 1 1 2

FingerprintsVector;TopologicalAtomTriplets:AtomicInvariantsAtomTypes:MinDistance1:MaxDistance10;3096;NumericalValues;IDsAndValuesString;C.X1
.BO1.H3-D1-C.X1.BO1.H3-D1-C.X3.BO3.H1-D2 C.X1.BO1.H3-D1-C.X2.BO2.H2-D1
O-C.X3.BO4-D9 C.X1.BO1.H3-D1-C.X2.BO2.H2-D3-N.X3.BO3-D4 C.X1.BO1.H3-D1

```
-C.X2.BO2.H2-D4-C.X2.BO2.H2-D5 C.X1.BO1.H3-D1-C.X2.BO2.H2-D6-C.X3....;
1 2 2 2 2 2 2 8 8 4 8 4 4 2 2 2 2 4 2 2 2 2 1 2 2 4 4 4 2 2
2 4 4 4 8 4 4 2 4 4 4 2 4 4 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 8...
```

```
FingerprintsVector;TopologicalAtomTriplets:SYBYLAtomTypes:MinDistance1
:MaxDistance10;2332;NumericalValues;IDsAndValuesString;C.2-D1-C.2-D9-C
.3-D10 C.2-D1-C.2-D9-C.ar-D10 C.2-D1-C.3-D1-C.3-D2 C.2-D1-C.3-D10-C.3-
D9 C.2-D1-C.3-D2-C.3-D3 C.2-D1-C.3-D2-C.ar-D3 C.2-D1-C.3-D3-C.3-D4 C.2
-D1-C.3-D3-N.ar-D4 C.2-D1-C.3-D3-O.3-D2 C.2-D1-C.3-D4-C.3-D5 C.2-D1-C.
3-D5-C.3-D6 C.2-D1-C.3-D5-O.3-D4 C.2-D1-C.3-D6-C.3-D7 C.2-D1-C.3-D7...
```

```
FingerprintsVector;TopologicalPharmacophoreAtomPairs:ArbitrarySize:Min
Distance1:MaxDistance10;54;NumericalValues;IDsAndValuesString;H-D1-H H
-D1-NI HBA-D1-NI HBD-D1-NI H-D2-H H-D2-HBA H-D2-HBD HBA-D2-HBA HBA-D2-
HBD H-D3-H H-D3-HBA H-D3-HBD H-D3-NI HBA-D3-NI HBD-D3-NI H-D4-H H-D4-H
BA H-D4-HBD HBA-D4-HBA HBA-D4-HBD HBD-D4-HBD H-D5-H H-D5-HBA H-D5-...;
18 1 2 1 22 12 8 1 2 18 6 3 1 1 1 22 13 6 5 7 2 28 9 5 1 1 1 36 16 10
3 4 1 37 10 8 1 35 10 9 3 3 1 28 7 7 4 18 16 12 5 1 2 1
```

```
FingerprintsVector;TopologicalPharmacophoreAtomPairs:FixedSize:MinDist
ance1:MaxDistance10;150;OrderedNumericalValues;ValuesString;18 0 0 1 0
0 0 2 0 0 1 0 0 0 0 22 12 8 0 0 1 2 0 0 0 0 0 0 0 0 18 6 3 1 0 0 0 1
0 0 1 0 0 0 0 22 13 6 0 0 5 7 0 0 2 0 0 0 0 0 28 9 5 1 0 0 0 1 0 0 1 0
0 0 0 36 16 10 0 0 3 4 0 0 1 0 0 0 0 0 37 10 8 0 0 0 0 1 0 0 0 0 0 0
0 35 10 9 0 0 3 3 0 0 1 0 0 0 0 0 28 7 7 4 0 0 0 0 0 0 0 0 0 0 0 18...
```

```
FingerprintsVector;TopologicalPharmacophoreAtomTriplets:ArbitrarySize:
MinDistance1:MaxDistance10;696;NumericalValues;IDsAndValuesString;Ar1-
Ar1-Ar1 Ar1-Ar1-H1 Ar1-Ar1-HBA1 Ar1-Ar1-HBD1 Ar1-H1-H1 Ar1-H1-HBA1 Ar1
-H1-HBD1 Ar1-HBA1-HBD1 H1-H1-H1 H1-H1-HBA1 H1-H1-HBD1 H1-HBA1-HBA1 H1-
HBA1-HBD1 H1-HBA1-NI1 H1-HBD1-NI1 HBA1-HBA1-NI1 HBA1-HBD1-NI1 Ar1-...;
46 106 8 3 83 11 4 1 21 5 3 1 2 2 1 1 1 100 101 18 11 145 132 26 14 23
28 3 3 5 4 61 45 10 4 16 20 7 5 1 3 4 5 3 1 1 1 1 5 4 2 1 2 2 2 1 1 1
119 123 24 15 185 202 41 25 22 17 3 5 85 95 18 11 23 17 3 1 1 6 4 ...
```

```
FingerprintsVector;TopologicalPharmacophoreAtomTriplets:FixedSize:MinD
istance1:MaxDistance10;2692;OrderedNumericalValues;ValuesString;46 106
8 3 0 0 83 11 4 0 0 0 1 0 0 0 0 0 0 0 0 21 5 3 0 0 1 2 2 0 0 1 0 0 0
0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 100 101 18 11 0 0 145 132 26
14 0 0 23 28 3 3 0 0 5 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 61 45 10 4 0
0 16 20 7 5 1 0 3 4 5 3 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 5 ...
```

METHODS

new

```
$NewFingerprintsFPFileIO = new FileIO::FingerprintsFPFileIO(%IOParameters);
```

Using specified *IOParameters* names and values hash, new method creates a new object and returns a reference to a newly created FingerprintsFPFileIO object. By default, the following properties are initialized during *Read* mode:

```
Name = '';
Mode = 'Read';
Status = 0;
FingerprintsStringMode = 'AutoDetect';
ValidateData = 1;
DetailLevel = 1;
```

During *Write* mode, the following properties get initialize by default:

```
FingerprintsStringMode = undef;

BitStringFormat = HexadecimalString;
```

```
BitsOrder = Ascending;
```

```
VectorStringFormat = NumericalValuesString or ValuesString;
```

Examples:

```
$NewFingerprintsFPFileIO = new FileIO::FingerprintsFPFileIO(
    'Name' => 'Sample.fpf',
    'Mode' => 'Read',
    'FingerprintsStringMode' =>
        'AutoDetect');

$NewFingerprintsFPFileIO = new FileIO::FingerprintsFPFileIO(
    'Name' => 'Sample.fpf',
    'Mode' => 'Write',
    'FingerprintsStringMode' =>
        'FingerprintsBitVectorString',
    'Overwrite' => 1,
    'BitStringFormat' => 'HexadecimalString',
    'BitsOrder' => 'Ascending');

$NewFingerprintsFPFileIO = new FileIO::FingerprintsFPFileIO(
    'Name' => 'Sample.fpf',
    'Mode' => 'Write',
    'FingerprintsStringMode' =>
        'FingerprintsVectorString',
    'Overwrite' => 1,
    'VectorStringFormat' => 'IDsAndValuesString');
```

GetFingerprints

```
$FingerprintsObject = $FingerprintsFPFileIO->GetFingerprints();
```

Returns FingerprintsObject generated for current data line using fingerprints bit-vector or vector string data. The fingerprints object corresponds to any of the supported fingerprints such as PathLengthFingerprints, ExtendedConnectivity, and so on.

GetFingerprintsString

```
$FingerprintsString = $FingerprintsFPFileIO->GetFingerprintsString();
```

Returns FingerprintsString for current data line.

GetHeaderDataKeyValue

```
$KeyValue = $FingerprintsFPFileIO->GetHeaderDataKeyValue($Key);
```

Returns KeyValue of a data header **Key**.

GetHeaderDataKeys

```
@Keys = $FingerprintsFPFileIO->GetHeaderDataKeys();
$NumOfKeys = $FingerprintsFPFileIO->GetHeaderDataKeys();
```

Returns an array of data header Keys retrieved from data header section of fingerprints file. In scalar context, it returns number of keys.

GetHeaderDataKeysAndValues

```
%KeysAndValues = $FingerprintsFPFileIO->GetHeaderDataKeysAndValues();
```

Returns a hash of data header keys and values retrieved from data header section of fingerprints file.

GetPartialFingerprintsString

```
$FingerprintsString = $FingerprintsFPFileIO->GetPartialFingerprintsString();
```

Returns partial FingerprintsString for current data line. It corresponds to fingerprints string specified

present in a line.

GetRequiredHeaderDataKeys

```
@Keys = $FingerprintsFPFileIO->GetRequiredHeaderDataKeys();
$NumOfKeys = $FingerprintsFPFileIO->GetRequiredHeaderDataKeys();
```

Returns an array of required data header Keys for a fingerprints file containing bit-vector or vector strings data. In scalar context, it returns number of keys.

GetRequiredHeaderDataKeysAndValues

```
%KeysAndValues = $FingerprintsFPFileIO->
    GetRequiredHeaderDataKeysAndValues();
```

Returns a hash of required data header keys and values for a fingerprints file containing bit-vector or vector strings data

IsFingerprintsDataValid

```
$Status = $FingerprintsFPFileIO->IsFingerprintsDataValid();
```

Returns 1 or 0 based on whether FingerprintsObject is valid.

IsFingerprintsFPFile

```
$Status = $FingerprintsFPFileIO->IsFingerprintsFPFile($FileName);
$Status = FileIO::FingerprintsFPFileIO::IsFingerprintsFPFile($FileName);
```

Returns 1 or 0 based on whether *FileName* is a FP file.

IsFingerprintsFileDataValid

```
$Status = $FingerprintsFPFileIO->IsFingerprintsFileDataValid();
```

Returns 1 or 0 based on whether fingerprints file contains valid fingerprints data.

IsHeaderDataKeyPresent

```
$Status = $FingerprintsFPFileIO->IsHeaderDataKeyPresent($Key);
```

Returns 1 or 0 based on whether data header *Key* is present in data header section of a FP file.

Next or Read

```
$FingerprintsFPFileIO = $FingerprintsFPFileIO->Next();
$FingerprintsFPFileIO = $FingerprintsFPFileIO->Read();
```

Reads next available fingerprints line in FP file, processes the data, generates appropriate fingerprints object, and returns FingerprintsFPFileIO. The generated fingerprints object is available using method GetFingerprints.

SetBitStringFormat

```
$FingerprintsFPFileIO->SetBitStringFormat($Format);
```

Sets bit string *Format* for fingerprints bit-vector string data in a FP file and returns FingerprintsFPFileIO. Possible values for BitStringFormat: *BinaryString* or *HexadecimalString*.

SetBitsOrder

```
$FingerprintsFPFileIO->SetBitsOrder($BitsOrder);
```

Sets *BitsOrder* for fingerprints bit-vector string data in a FP file and returns FingerprintsFPFileIO. Possible values for BitsOrder: *Ascending* or *Descending*.

SetCompoundID

```
$FingerprintsFPFileIO->SetCompoundID($ID);
```

Sets compound ID for current data line and returns FingerprintsFPFileIO. Spaces are not allowed in compound IDs.

SetDetailLevel

```
$FingerprintsFPFileIO->SetDetailLevel($Level);
```

Sets details *Level* for generating diagnostics messages during FP file processing and returns FingerprintsFPFileIO. Possible values: *Positive integers*.

SetFingerprints

```
$FingerprintsFPFileIO->SetFingerprints($FingerprintsObject);
```

Sets *FingerprintsObject* for current data line and returns FingerprintsFPFileIO.

SetFingerprintsString

```
$FingerprintsFPFileIO->SetFingerprintsString($FingerprintsString);
```

Sets *FingerprintsString* for current data line and returns FingerprintsFPFileIO.

SetFingerprintsStringMode

```
$FingerprintsFPFileIO->SetFingerprintsStringMode($Mode);
```

Sets *FingerprintsStringMode* for FP file and returns FingerprintsFPFileIO. Possible values: *AutoDetect*, *FingerprintsBitVectorString*, or *FingerprintsVectorString*

SetPartialFingerprintsString

```
$FingerprintsFPFileIO->SetPartialFingerprintsString($PartialString);
```

Sets *PartialFingerprintsString* for current data line and returns FingerprintsFPFileIO.

SetVectorStringFormat

```
$FingerprintsFPFileIO->SetVectorStringFormat($Format);
```

Sets *VectorStringFormat* for FP file and returns FingerprintsFPFileIO. Possible values: *IDsAndValuesString*, *IDsAndValuesPairsString*, *ValuesAndIDsString*, *ValuesAndIDsPairsString*.

WriteFingerprints

```
$FingerprintsFPFileIO->WriteFingerprints($FingerprintsObject,  
                                         $CompoundID);
```

Writes fingerprints string generated from *FingerprintsObject* object and other data including *CompoundID* to FP file and returns FingerprintsFPFileIO.

WriteFingerprintsString

```
$FingerprintsFPFileIO->WriteFingerprints($FingerprintsString,  
                                         $CompoundID);
```

Writes *FingerprintsString* and other data including *CompoundID* to FP file and returns FingerprintsFPFileIO.

Caveats:

- o FingerprintsStringMode, BitStringFormat, BitsOrder, VectorStringFormat values are ignored during writing of fingerprints and it's written to the file as it is.
- o FingerprintsString is a regular fingerprints string as oppose to a partial fingerprints string.

AUTHOR

Manish Sud <msud@san.rr.com>

SEE ALSO

FingerprintsSDFFileIO.pm, FingerprintsTextFileIO.pm

COPYRIGHT

Copyright (C) 2020 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.