pyOpenMS

OpenMS Development Team

pyOpenMS is a set of Python bindings of the C++ OpenMS library. It allows to access a large number of objects (350+) and functions (3900+) of the C++ code directly from Python. The main functions of the library are explained in the first section of this manual. A list of all wrapped functions can be found in the appendix of this manual.

Since all functions in Python directly call C++ and their function signature usually corresponds to the one in C++, the OpenMS documentation is for most cases the most complete and up-to-date reference also and applies directly to pyOpenMS. In this manual, only differences to the existing documentation will be highlighted and some general usecases will be explained. The link to the documentation of the latest release can be found here: `http://open-ms.sourceforge.net/documentation/`.

Contents

The following section will explain the most important functions of pyOpenMS in more detail with full examples of Python code that can be directly executed.

1 File Input/Output

pyOpenMS supports file input and output for various formats. These formats include `DTA2D, DTA, EDTA, FeatureXML, Kroenik, MzData, MzIdentML, IdXML, mzML, mzXML, PepXML, ProtXML, TraML, XTandemXML`.

1.1 Common Pattern

Most file format objects follow the following idiome

```
from pyopenms import *
file = FileObject()
exp = MSExperiment()
file.load(filename_in, exp)
# process data
file.store(filename_in, exp)
```

where `FileObject` can be any of `DTA2DFile, DTAFile, mzXMLFile, mzMLFile ...` (note that the above code will not work directly since you will need to use a specific file implementation, see below).

For `EDTAFile`, you must load and store a `ConsensusMap` instead of an `MSExperiment`. for `FeatureXMLFile`, you must load and store a `FeatureMap` instead of an `MSExperiment`.

1.2 IdXML

```
from pyopenms import *
id_file = IdXMLFile()
filename_in = "input.IdXML"
filename_out = "output.IdXML"
protein_ids = []
peptide_ids = []
id_file.load(filename_in, protein_ids, peptide_ids)
# process
id_file.store(filename_out, protein_ids, peptide_ids)
```

see also Section 6.147.

1.3 PepXML

```
from pyopenms import *
id_file = PepXMLFile()
filename_in = "input.pep.xml"
```

```
filename_out = "output.pep.xml"
protein_ids = []
peptide_ids = []
id_file.load(filename_in, protein_ids, peptide_ids)
# process
id_file.store(filename_out, protein_ids, peptide_ids)
```

`PepXML` also supports loading with an additional parameter specificing an MSExperiment which contains the retention time corresponding to the peptide hits (since these may not be stored in a pep.xml file).

see also Section 6.304.

1.4 ProtXML

```
from pyopenms import *
id_file = ProtXMLFile()
filename_in = "input.prot.xml"
protein_id = ProteinIdentification()
peptide_id = PeptideIdentification()
id_file.load(filename_in, protein_id, peptide_id)
# process
# storing not supported
```

`ProtXML` currently only supports loading of data.

1.5 MzIdentML

```
from pyopenms import *
id_file = MzIdentMLFile()
filename_in = "input.mzid"
filename_out = "output.mzid"
identification = Identification()
id_file.load(filename_in, identification)
# process
id_file.store(filename_out, identification)
```

Alternatively, MzIdentMLFile also provides a function to load (but not store) data equivalent to IdXML using two empty vectors that will be filled with `ProteinIdentification` and `PeptideIdentification` objects.

1.6 TraML

```
from pyopenms import *
tramlfile = TraMLFile()
filename_in = "input.TraML"
targeted_exp = TargetedExperiment()
```

```
idtramlfile.load(filename_in, targeted_exp)
# process
tramlfilefile.store(filename_out, targeted_exp)
```

see also Section 6.399.

1.7 MzML

```
from pyopenms import *
file = MzMLFile()
exp = MSExperiment()
filename_in = "input.mzML"
filename_out = "output.mzML"
file.load(filename_in, exp)
# process
file.store(filename_in, exp)
```

see also Section 6.257.

2 Parameter Handling

Paramter handling in OpenMS and pyOpenMS is usually implemented through inheritance from `DefaultParamHandler` and allow access to parameters through the `Param` object. This means, the classes implement the methods `getDefaults`, `getParameters`, `setParameters` which allows access to the default parameters, the current parameters and allows to set the parameters.

The Param file that is returned can be manipulated through the `setValue` and `getValue` methods (the `existsts` method can be used to check for existance of a key). Using the `getDescription` method, it is possible to get a help-text for each parameter value in an interactive session without consulting the documentation.

3 Signal Processing and Filter

Most signal processing algorithms follow a similar pattern in OpenMS.

```
filter = FilterObject()
exp = MSExperiment()
# populate exp
filter.filterExperiment(exp)
```

Since they work on a single `MSExperiment` object, little input is needed to execute a filter directly on the data. Examples of filters that follow this pattern are `GaussFilter, SavitzkyGolayFilter` as well as the spectral filters `BernNorm, MarkerMower, NLargest, Normalizer, ParentPeakMower, Scaler, SpectraMerger, SqrtMower, ThresholdMower, WindowMower`.

4 Complex algorithmic tools

More complex algorithmic tools require a short explanation and usage:

4.1 iTRAQ Analyzer

The iTRAQ Analyzer uses internally two classes, the `ItraqChannelExtractor` and the `ItraqQuantifier` which are described in the following code snippet:

```
file = MzMLFile()
exp = MSExperiment()
file.load(filename_in, exp)
consensus_map_raw = ConsensusMap()
consensus_map_quant = ConsensusMap()

itraq_type = ITRAQ_TYPES.FOURPLEX # other options: EIGHTPLEX, TMT_SIXPLEX
extract_param = ItraqChannelExtractor().getDefaults()
```

```
itraq_ce = ItraqChannelExtractor(itraq_type, extract_param)

# extract raw signals
itraq_ce.run(exp, consensus_map_raw)

# do normalization
quant_param = ItraqQuantifier().getDefaults()
itraq_quant = ItraqQuantifier(itraq_type, quant_param)

itraq_quant.run(consensus_map_raw, consensus_map_quant)

outfile = ConsensusXMLFile()
oufile.store(filename_out, consensus_map_quant)
```

4.2 Centroided FeatureFinder

The FeatureFinder for centroided data is called `FeatureFinderAlgorithmPicked` in OpenMS.

```
# set input_path and out_path
seeds = FeatureMap()

fh = MzMLFile()
options = PeakFileOptions()
options.setMSLevels([1,1])
fh.setOptions(options)
input_map = MSExperiment()
fh.load(input_path, input_map)
input_map.updateRanges()

ff = FeatureFinder()
ff.setLogType(LogType.CMD)

# Run the feature finder
features = FeatureMap()
name = FeatureFinderAlgorithmPicked.getProductName()
params = FeatureFinder().getParameters(name)
ff.run(name, input_map, features, params, seeds)

features.setUniqueIds()
fh = FeatureXMLFile()
fh.store(out_path, features)
```

4.3 OpenSwathAnalyzer

The OpenSwathAnalyzer calls internally an object called MRMFeatureFinderScoring (since it does feature finding based on a scoring approach). It takes as input the chromatograms

and the targeted library (transition library) in TraML format. Furthermore, it also takes a transformation description and a Swath file as optional arguments.

```
# load chromatograms
chromatograms = pyopenms.MSExperiment()
fh = pyopenms.FileHandler()
fh.loadExperiment("infile.mzML", chromatograms)

# load TraML file
targeted = pyopenms.TargetedExperiment();
tramlfile = pyopenms.TraMLFile();
tramlfile.load("tramlfile.TraML", targeted);

# Create empty files as input and finally as output
empty_swath = pyopenms.MSExperiment()
trafo = pyopenms.TransformationDescription()
output = pyopenms.FeatureMap();

# set up OpenSwath analyzer (featurefinder) and run
featurefinder = pyopenms.MRMFeatureFinderScoring()
featurefinder.pickExperiment(chromatograms, output, targeted, trafo, empty_swath)

# Store outfile
featurexml = pyopenms.FeatureXMLFile()
featurexml.store("outfile.featureXML", output)
```

5 Iterators

Several core-OpenMS objects have been adapted to allow iteration in a native way in Python. These objects are currently `ConsensusMap, FeatureMap, MSExperiment, MSSpectrum` and `MSChromatogram`. They thus allow the following syntax:

```
for spectrum in ms_experiment:
  for peak in spectrum:
    # process an individual peak
    pass
```

6 Appendix

In this appendix, a complete list of all wrapped functions is given, ordered by class. Note that not all C++ functions are wrapped in Python and the following documentation indicates which do have wrappers and can be used directly from Python. The following appendix also does not contain actual documentation of the functionality, please use the link to the OpenMS documentation to find up-to-date documentation on each class and member function.

6.1 AAIndex

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/AAIndex.h>" namespace "OpenMS":

  cdef cppclass AAIndex "OpenMS::AAIndex":

    AAIndex(AAIndex)  #wrap-ignore

    double aliphatic(char aa)
    double acidic(char aa)
    double basic(char aa)
    double polar(char aa)
    double getKHAG800101(char aa)
    double getVASM830103(char aa)
    double getNADH010106(char aa)
    double getNADH010107(char aa)
    double getWILM950102(char aa)
    double getROBB760107(char aa)
    double getOOBM850104(char aa)
    double getFAUJ880111(char aa)
    double getFINA770101(char aa)
    double getARGP820102(char aa)
    double calculateGB(AASequence &seq, double T)
```

6.2 AASequence

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/AASequence.h>" namespace "OpenMS":

  cdef cppclass AASequence:
    # wrap-hash:
    #   toString().c_str()

    AASequence()
    AASequence(AASequence)  # wrap-ignore

    AASequence operator+(AASequence)
    AASequence iadd(AASequence)    # wrap-as:operator+=

    bool empty()
```

```
String toString()

String toUnmodifiedString()

String toBracketString(libcpp_vector[String] fixed_modifications)

void setModification(Size index, String modification)

void setNTerminalModification(String modification)

String getNTerminalModificationName()

void setCTerminalModification(String modification)

String getCTerminalModificationName()

Residue getResidue(Size index)

EmpiricalFormula getFormula(ResidueType type_, Int charge)

double getAverageWeight(ResidueType type_, Int charge)

double getMonoWeight(ResidueType type_, Int charge)

Size size()

AASequence getPrefix(Size index)

AASequence getSuffix(Size index)

AASequence getSubsequence(Size index, UInt number)

void getAAFrequencies(Map[String, size_t])  # wrap-ignore

bool has(Residue residue)

bool hasSubsequence(AASequence peptide)

bool hasPrefix(AASequence peptide)

bool hasSuffix(AASequence peptide)

bool hasNTerminalModification()

bool hasCTerminalModification()

bool isModified()
```

```
#  wrap static methods
cdef extern from "<OpenMS/CHEMISTRY/AASequence.h>" namespace "OpenMS::AASequence":

    AASequence fromString(String s, bool permissive)   # wrap-attach:AASequence
```

6.3 AScore

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/ID/AScore.h>" namespace "OpenMS":

  cdef cppclass AScore:

    AScore()
    AScore(AScore)  # wrap-ignore

    PeptideHit compute(PeptideHit & hit,
               MSSpectrum[Peak1D] & real_spectrum,
               double fragment_mass_tolerance,
               bool fragment_mass_unit_ppm)

  cdef cppclass ProbablePhosphoSites:

    ProbablePhosphoSites()
    ProbablePhosphoSites(ProbablePhosphoSites)  # wrap-ignore

    Size first
    Size second
    Size seq_1
    Size seq_2
    Size peak_depth
    Size AScore
```

6.4 AccurateMassSearchEngine

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/ID/AccurateMassSearchEngine.h>" namespace\
       "OpenMS":

  cdef cppclass AccurateMassSearchEngine(DefaultParamHandler,ProgressLogger) :
    # wrap-inherits:
```

```
    #  DefaultParamHandler
    #  ProgressLogger
    AccurateMassSearchEngine()
    AccurateMassSearchEngine(AccurateMassSearchEngine)  #wrap-ignore

    void queryByMZ(double observed_mz, Int observed_charge, String ion_mode,
             libcpp_vector[ AccurateMassSearchResult ] & )

    void queryByFeature(Feature feature, Size feature_index, String ion_mode,
              libcpp_vector[ AccurateMassSearchResult ] & )

    void queryByConsensusFeature(ConsensusFeature cfeat, Size cf_index,
                   Size number_of_maps, String ion_mode,
                   libcpp_vector[AccurateMassSearchResult]& results)

    void run(FeatureMap & , MzTab & )
    void run(ConsensusMap&, MzTab&)

    void init()
```

6.5 AccurateMassSearchResult

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/ID/AccurateMassSearchEngine.h>" namespace\
       "OpenMS":

  cdef cppclass AccurateMassSearchResult "OpenMS::AccurateMassSearchResult":
    AccurateMassSearchResult()
    AccurateMassSearchResult(AccurateMassSearchResult)
    double getObservedMZ()
    void setObservedMZ(double & m)
    double getCalculatedMZ()
    void setCalculatedMZ(double & m)
    double getQueryMass()
    void setQueryMass(double & m)
    double getFoundMass()
    void setFoundMass(double & m)
    double getCharge()
    void setCharge(double & ch)
    double getMZErrorPPM()
    void setMZErrorPPM(double & ppm)
    double getObservedRT()
    void setObservedRT(double & rt)
    double getObservedIntensity()
    void setObservedIntensity(double & intensity)
```

```
    double getMatchingIndex()
    void setMatchingIndex(double & idx)
    String getFoundAdduct()
    void setFoundAdduct(String & add)
    String getFormulaString()
    void setEmpiricalFormula(String & ep)
    libcpp_vector[ String ] getMatchingHMDBids()
    void setMatchingHMDBids(libcpp_vector[ String ] & match_ids)
    double getIsotopesSimScore()
    void setIsotopesSimScore(double & sim_score)
    libcpp_vector[double] getIndividualIntensities()
    void setIndividualIntensities(libcpp_vector[double])
    Size getSourceFeatureIndex()
    void setSourceFeatureIndex(Size)
    libcpp_vector[ double] getMasstraceIntensities()
    void setMasstraceIntensities(libcpp_vector[ double ] & )
```

6.6 Acquisition

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/Acquisition.h>" namespace "OpenMS":

  cdef cppclass Acquisition "OpenMS::Acquisition":
    Acquisition()
    Acquisition(Acquisition)
    bool operator==(Acquisition &rhs)
    bool operator!=(Acquisition &rhs)
    String  getIdentifier()
    void setIdentifier(String &identifier)
```

6.7 AcquisitionInfo

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/AcquisitionInfo.h>" namespace "OpenMS":

  cdef cppclass AcquisitionInfo:

    AcquisitionInfo()
    AcquisitionInfo(AcquisitionInfo)

    bool operator==(AcquisitionInfo)
```

```
    bool operator!=(AcquisitionInfo)

    String getMethodOfCombination()
    void setMethodOfCombination(String method)
```

6.8 Adduct

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/DATASTRUCTURES/Adduct.h>" namespace "OpenMS":

  cdef cppclass Adduct:

    Adduct()
    Adduct(Adduct)

    Adduct(Int charge)
    Adduct(Int charge, Int amount, double singleMass, String formula, double\
       log_prob, double rt_shift, String label)

    Int getCharge()

    void setCharge(Int charge)

    Int getAmount()
    void setAmount(Int amount)

    double getSingleMass()
    void setSingleMass(double singleMass)

    double getLogProb()
    void setLogProb(double log_prob)

    String getFormula()
    void setFormula(String formula)

    double getRTShift()
    String getLabel()
```

6.9 AnnotationStatistics

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/KERNEL/FeatureMap.h>" namespace "OpenMS":
```

```
  cdef cppclass AnnotationStatistics "OpenMS::AnnotationStatistics":
    AnnotationStatistics()
    AnnotationStatistics(AnnotationStatistics)
    libcpp_vector[ size_t ] states
    bool operator==(AnnotationStatistics & rhs)
```

6.10 Attachment

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/QcMLFile.h>" namespace "OpenMS::QcMLFile":

  cdef cppclass Attachment "OpenMS::QcMLFile::Attachment":
    Attachment()
    Attachment(Attachment)
    String name
    String id
    String value
    String cvRef
    String cvAcc
    String unitRef
    String unitAcc
    String binary
    String qualityRef
    libcpp_vector[ String ] colTypes
    libcpp_vector[ libcpp_vector[ String ] ] tableRows # wrap-ignore
    bool operator==(Attachment &rhs)
    bool operator<(Attachment &rhs)
    bool operator>(Attachment &rhs)
    String toXMLString(UInt indentation_level)
    String toCSVString(String separator)
```

6.11 AverageLinkage

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/COMPARISON/CLUSTERING/AverageLinkage.h>" namespace\
       "OpenMS":

  cdef cppclass AverageLinkage "OpenMS::AverageLinkage":
    AverageLinkage()
    AverageLinkage(AverageLinkage)
```

```
    String getProductName()
```

6.12 BSpline2d

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/MATH/MISC/BSpline2d.h>" namespace "OpenMS":

  cdef cppclass BSpline2d:

    BSpline2d(libcpp_vector[double] x, libcpp_vector[double] y,
          double wave_length, BoundaryCondition boundary_condition,
          Size num_nodes)

    bool solve(libcpp_vector[double] y)

    double eval(double x)

    double derivative(double x)

    bool ok()

    void debug(bool enable)

cdef extern from "<OpenMS/MATH/MISC/BSpline2d.h>" namespace "OpenMS::BSpline2d":

  cdef enum BoundaryCondition:
    BC_ZERO_ENDPOINTS, BC_ZERO_FIRST, BC_ZERO_SECOND
```

6.13 Base64

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/Base64.h>" namespace "OpenMS":

  cdef cppclass Base64 "OpenMS::Base64":
    Base64()
    Base64(Base64)  #wrap-ignore

    void encode(libcpp_vector[ double ] & in_, ByteOrder to_byte_order, String &out,\
       bool zlib_compression)
    void decode(String & in_, ByteOrder from_byte_order, libcpp_vector[ double ]\
```

```
       &out, bool zlib_compression)
    void encodeIntegers(libcpp_vector[ int ] & in_, ByteOrder to_byte_order, String\
       &out, bool zlib_compression)
    void decodeIntegers(String & in_, ByteOrder from_byte_order, libcpp_vector[ int\
       ] &out, bool zlib_compression)

    void encodeStrings(libcpp_vector[ String ] & in_, String &out, bool\
       zlib_compression)
    void decodeStrings(String & in_, libcpp_vector[ String ] &out, bool\
       zlib_compression)

cdef extern from "<OpenMS/FORMAT/Base64.h>" namespace "OpenMS::Base64":
  cdef enum ByteOrder "OpenMS::Base64::ByteOrder":
    #wrap-attach:
    #  Base64
    BYTEORDER_BIGENDIAN
    BYTEORDER_LITTLEENDIAN
```

6.14 BaseFeature

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/KERNEL/BaseFeature.h>" namespace "OpenMS":

  cdef cppclass BaseFeature(UniqueIdInterface):
    # wrap-inherits:
    #  UniqueIdInterface

    BaseFeature()
    BaseFeature(BaseFeature &)

    float getQuality()
    void setQuality(float q)

    float getWidth()
    void setWidth(float q)

    Int getCharge()
    void setCharge(Int q)
    AnnotationState getAnnotationState()

    libcpp_vector[PeptideIdentification] getPeptideIdentifications()
    void setPeptideIdentifications(libcpp_vector[PeptideIdentification] & peptides)

    bool operator==(BaseFeature)
```

```
    bool operator!=(BaseFeature)

    void getKeys(libcpp_vector[String] & keys)
    void getKeys(libcpp_vector[unsigned int] & keys)  # wrap-as:getKeysAsIntegers
    DataValue getMetaValue(unsigned int)
    DataValue getMetaValue(String)
    void setMetaValue(unsigned int, DataValue)
    void setMetaValue(String, DataValue)
    bool metaValueExists(String)
    bool metaValueExists(unsigned int)
    void removeMetaValue(String)
    void removeMetaValue(unsigned int)
    void clearMetaInfo()

cdef extern from "<OpenMS/KERNEL/BaseFeature.h>" namespace "OpenMS::BaseFeature":

  cdef enum AnnotationState "OpenMS::BaseFeature::AnnotationState":
    FEATURE_ID_NONE
    FEATURE_ID_SINGLE
    FEATURE_ID_MULTIPLE_SAME
    FEATURE_ID_MULTIPLE_DIVERGENT
    SIZE_OF_ANNOTATIONSTATE
```

6.15 BaseGroupFinder

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/MAPMATCHING/BaseGroupFinder.h>" namespace\
       "OpenMS":

  cdef cppclass BaseGroupFinder(DefaultParamHandler,ProgressLogger) :
    # wrap-ignore
    # ABSTRACT class
    # wrap-inherits:
    #  DefaultParamHandler
    #  ProgressLogger
    BaseGroupFinder()
    BaseGroupFinder(BaseGroupFinder)  #wrap-ignore

    void registerChildren()
```

6.16 BernNorm

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/TRANSFORMERS/BernNorm.h>" namespace "OpenMS":

  cdef cppclass BernNorm(DefaultParamHandler):
    # wrap-inherits:
    #  DefaultParamHandler

    BernNorm()
    BernNorm(BernNorm)    #wrap-ignore

    void filterSpectrum(MSSpectrum[Peak1D] & spec)
    void filterPeakSpectrum(MSSpectrum[Peak1D] & spec)
    void filterPeakMap(MSExperiment[Peak1D, ChromatogramPeak] & exp)
```

6.17 BiGaussFitter1D

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/BiGaussFitter1D.h>"\
       namespace "OpenMS":

  cdef cppclass BiGaussFitter1D "OpenMS::BiGaussFitter1D":
    BiGaussFitter1D()
    BiGaussFitter1D(BiGaussFitter1D)
    String getProductName()
```

6.18 BiGaussModel

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/BiGaussModel.h>" namespace\
       "OpenMS":

  cdef cppclass BiGaussModel "OpenMS::BiGaussModel":
    BiGaussModel()
    BiGaussModel(BiGaussModel)
    void setOffset(double offset)
    void setSamples()
    double getCenter()
    String getProductName()
```

6.19 BinnedSpectrum

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/COMPARISON/SPECTRA/BinnedSpectrum.h>" namespace "OpenMS":

  cdef cppclass BinnedSpectrum:

    BinnedSpectrum()
    BinnedSpectrum(BinnedSpectrum)
    BinnedSpectrum(float size, UInt spread, MSSpectrum[Peak1D] ps)
    bool operator==(BinnedSpectrum & rhs)
    bool operator!=(BinnedSpectrum & rhs)
    bool operator==(MSSpectrum[Peak1D] & rhs)
    bool operator!=(MSSpectrum[Peak1D] & rhs)
    double getBinSize()
    UInt getBinSpread()
    UInt getBinNumber()
    UInt getFilledBinNumber()
    void setBinSize(double s)
    void setBinSpread(UInt s)
    void setBinning()
    bool checkCompliance(BinnedSpectrum & bs)
    MSSpectrum[Peak1D] getRawSpectrum()
```

6.20 CVMappingFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/CVMappingFile.h>" namespace "OpenMS":

  cdef cppclass CVMappingFile:

    CVMappingFile()
    void load(String & filename, CVMappings & cv_mappings, bool strip_namespaces)
```

6.21 CVMappingRule

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/DATASTRUCTURES/CVMappingRule.h>" namespace "OpenMS":

  cdef cppclass CVMappingRule:

    CVMappingRule()
    CVMappingRule(CVMappingRule)

    void setIdentifier(String identifier)

    String getIdentifier()

    void setElementPath(String element_path)

    String getElementPath()

    void setRequirementLevel(RequirementLevel level)

    RequirementLevel getRequirementLevel()

    void setCombinationsLogic(CombinationsLogic combinations_logic)

    CombinationsLogic getCombinationsLogic()

    void setScopePath(String path)

    String getScopePath()

    void setCVTerms(libcpp_vector[CVMappingTerm] cv_terms)

    libcpp_vector[CVMappingTerm] getCVTerms()

    void addCVTerm(CVMappingTerm cv_terms)

    bool operator==(CVMappingRule rhs)

    bool operator!=(CVMappingRule rhs)

cdef extern from "<OpenMS/DATASTRUCTURES/CVMappingRule.h>" namespace\
       "OpenMS::CVMappingRule":

  cdef enum RequirementLevel:
    # wrap-attach:
    #   CVMappingRule
    MUST = 0,
    SHOULD = 1,
    MAY = 2

  cdef enum CombinationsLogic:
```

```
    # wrap-attach:
    #   CVMappingRule
    OR = 0,
    AND = 1,
    XOR = 2
```

6.22 CVMappingTerm

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/DATASTRUCTURES/CVMappingTerm.h>" namespace "OpenMS":

  cdef cppclass CVMappingTerm:

    CVMappingTerm()
    CVMappingTerm(CVMappingTerm)

    void setAccession(String accession)

    String getAccession()

    void setUseTermName(bool use_term_name)

    bool getUseTermName()

    void setUseTerm(bool use_term)

    bool getUseTerm()

    void setTermName(String term_name)

    String getTermName()

    void setIsRepeatable(bool is_repeatable)

    bool getIsRepeatable()

    void setAllowChildren(bool allow_children)

    bool getAllowChildren()

    void setCVIdentifierRef(String cv_identifier_ref)

    String getCVIdentifierRef()

    bool operator==(CVMappingTerm rhs)
```

```
    bool operator!=(CVMappingTerm rhs)
```

6.23 CVMappings

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/DATASTRUCTURES/CVMappings.h>" namespace "OpenMS":

  cdef cppclass CVMappings:

    CVMappings()
    CVMappings(CVMappings)
    void setMappingRules(libcpp_vector[ CVMappingRule ] &cv_mapping_rules)
    libcpp_vector[ CVMappingRule ]  getMappingRules()
    void addMappingRule(CVMappingRule &cv_mapping_rule)
    void setCVReferences(libcpp_vector[ CVReference ] &cv_references)
    libcpp_vector[ CVReference ]  getCVReferences()
    void addCVReference(CVReference &cv_reference)
    bool hasCVReference(String &identifier)
```

6.24 CVReference

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/DATASTRUCTURES/CVReference.h>" namespace "OpenMS":

  cdef cppclass CVReference "OpenMS::CVReference":
    CVReference()
    CVReference(CVReference)
    void setName(String &name)
    String  getName()
    void setIdentifier(String &identifier)
    String  getIdentifier()
    bool operator==(CVReference &rhs)
    bool operator!=(CVReference &rhs)
```

6.25 CVTerm

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/CVTerm.h>" namespace "OpenMS":

  cdef cppclass CVTerm:
     CVTerm()
     CVTerm(CVTerm)

     bool operator==(CVTerm)

     void setAccession(String accession)
     String getAccession()

     void setName(String name)
     String getName()

     void setCVIdentifierRef(String cv_id_ref)
     String getCVIdentifierRef()

     DataValue getValue()
     void setValue(DataValue value)

     void setUnit(Unit & unit)
     Unit  getUnit()
     bool hasValue()
     bool hasUnit()

cdef extern from "<OpenMS/METADATA/CVTerm.h>" namespace "OpenMS::CVTerm":

  cdef cppclass Unit "OpenMS::CVTerm::Unit":
    Unit()
    Unit(Unit)
    String accession
    String name
    String cv_ref
    Unit(String & p_accession, String & p_name, String & p_cv_ref)
    bool operator==(Unit & rhs)
    bool operator!=(Unit & rhs)
```

6.26 CVTermList

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/CVTermList.h>" namespace "OpenMS":

  cdef cppclass CVTermList(MetaInfoInterface):
```

```
    # wrap-inherits:
    #  MetaInfoInterface
    ########################################################################
    # cython has a problem with inheritance of overloaded methods,
    # see eg Precursor.pxd

    CVTermList()
    CVTermList(CVTermList)

    void setCVTerms(libcpp_vector[CVTerm] & terms)
    void replaceCVTerm(CVTerm & term)

    void replaceCVTerms(libcpp_vector[CVTerm] cv_terms,
              String accession
               )

    void replaceCVTerms(Map[String, libcpp_vector[CVTerm] ] cv_term_map)

    void consumeCVTerms(Map[String, libcpp_vector[CVTerm] ] cv_term_map)

    Map[String, libcpp_vector[CVTerm] ] getCVTerms()
    void addCVTerm(CVTerm & term)

    bool operator==(CVTermList)
    bool operator!=(CVTermList)

    bool hasCVTerm(String accession)
    bool empty()
```

6.27 CachedmzML

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/CachedMzML.h>" namespace "OpenMS":

  cdef cppclass CachedmzML(ProgressLogger):
    # wrap-inherits:
    #   ProgressLogger

    CachedmzML()
    CachedmzML(CachedmzML)

    void writeMemdump(MSExperiment[Peak1D,ChromatogramPeak] exp, String out)
    void writeMetadata(MSExperiment[Peak1D,ChromatogramPeak] exp, String out_meta)
```

```
    void readMemdump(MSExperiment[Peak1D,ChromatogramPeak] exp, String filename)

    libcpp_vector[ streampos ]  getSpectraIndex()
    libcpp_vector[ streampos ]  getChromatogramIndex()
    void createMemdumpIndex(String filename)
```

6.28 CalibrationData

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/DATASTRUCTURES/CalibrationData.h>" namespace "OpenMS":

  cdef cppclass CalibrationData:

    CalibrationData()
    CalibrationData(CalibrationData &)
    double getMZ(Size)
    double getRT(Size)
    double getIntensity(Size)
    # libcpp_vector[RichPeak2D].iterator begin()  #\
       wrap-iter-begin:__iter__(RichPeak2D)
    # libcpp_vector[RichPeak2D].iterator end()  #\
       wrap-iter-end:__iter__(RichPeak2D)
    Size size()
    bool empty()
    void clear()
    void setUsePPM(bool)
    bool usePPM()
    void insertCalibrationPoint(double rt, double mz_obs, float intensity, double\
       mz_ref, double weight, int group)
    Size getNrOfGroups()
    double getError(Size)
    double getRefMZ(Size)
    double getWeight(Size)
    int getGroup(Size i)
    CalibrationData median(double, double)
    void sortByRT()

cdef extern from "<OpenMS/DATASTRUCTURES/CalibrationData.h>" namespace\
       "OpenMS::CalibrationData":

    StringList getMetaValues()  # wrap-attach:CalibrationData
```

6.29 ChargePair

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/DATASTRUCTURES/ChargePair.h>" namespace "OpenMS":

  cdef cppclass ChargePair:

    ChargePair()
    ChargePair(ChargePair)

    ChargePair(Size index0,
         Size index1,
         Int charge0,
         Int charge1,
         Compomer compomer,
         double mass_diff,
           bool active)

    Int getCharge(UInt pairID)
    void setCharge(UInt pairID, Int e)

    Size getElementIndex(UInt pairID)
    void setElementIndex(UInt pairID, Size e)

    Compomer getCompomer()
    void setCompomer( Compomer & compomer)

    double getMassDiff()
    void setMassDiff(double mass_diff)

    double getEdgeScore()
    void setEdgeScore(double score)

    bool isActive()
    void setActive( bool active)
```

6.30 ChromatogramExtractor

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/OPENSWATH/ChromatogramExtractor.h>" namespace\
       "OpenMS":

  cdef cppclass ChromatogramExtractor(ProgressLogger):
```

```
    # wrap-inherits:
    #  ProgressLogger

    ChromatogramExtractor()
    ChromatogramExtractor(ChromatogramExtractor)

    void extractChromatograms(MSExperiment[Peak1D, ChromatogramPeak] & input,
                  MSExperiment[Peak1D, ChromatogramPeak] & output,
                  TargetedExperiment & transition_exp,
                  double extract_window,
                  bool ppm,
                  TransformationDescription trafo,
                  double rt_extraction_window,
                  String filter)
```

6.31 ChromatogramExtractorAlgorithm

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/OPENSWATH/ChromatogramExtractorAlgorithm.h>"\
       namespace "OpenMS":

  cdef cppclass ChromatogramExtractorAlgorithm(ProgressLogger):
    # wrap-inherits:
    #  ProgressLogger

    ChromatogramExtractorAlgorithm()
    ChromatogramExtractorAlgorithm(ChromatogramExtractorAlgorithm)

    void extractChromatograms(
      shared_ptr[ SpectrumAccessOpenMS ] input,
      libcpp_vector[ shared_ptr[OSChromatogram] ] & output,
      libcpp_vector[ ExtractionCoordinates ] extraction_coordinates,
      double mz_extraction_window,
      bool ppm, String filter)

    void extractChromatograms(
      shared_ptr[ SpectrumAccessOpenMSCached ] input,
      libcpp_vector[ shared_ptr[OSChromatogram] ] & output,
      libcpp_vector[ ExtractionCoordinates ] extraction_coordinates,
      double mz_extraction_window,
      bool ppm, String filter)

    void extractChromatograms(
      shared_ptr[ SpectrumAccessOpenMSInMemory ] input,
```

```
      libcpp_vector[ shared_ptr[OSChromatogram] ] & output,
      libcpp_vector[ ExtractionCoordinates ] extraction_coordinates,
      double mz_extraction_window,
      bool ppm, String filter)

    void extractChromatograms(
      shared_ptr[ SpectrumAccessQuadMZTransforming ] input,
      libcpp_vector[ shared_ptr[OSChromatogram] ] & output,
      libcpp_vector[ ExtractionCoordinates ] extraction_coordinates,
      double mz_extraction_window,
      bool ppm, String filter)

cdef extern from "<OpenMS/ANALYSIS/OPENSWATH/ChromatogramExtractorAlgorithm.h>"\
       namespace "OpenMS::ChromatogramExtractorAlgorithm":

  cdef cppclass ExtractionCoordinates:

    ExtractionCoordinates()
    ExtractionCoordinates(ExtractionCoordinates)

    double mz # mz around which should be extracted
    double rt_start # rt start of extraction (in seconds)
    double rt_end # rt end of extraction (in seconds)
    libcpp_string id # identifier
```

6.32 ChromatogramPeak

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/KERNEL/ChromatogramPeak.h>" namespace\
       "OpenMS::ChromatogramPeak":

  ctypedef double IntensityType
  ctypedef double CoordinateType

cdef extern from "<OpenMS/KERNEL/ChromatogramPeak.h>" namespace "OpenMS":

  cdef cppclass ChromatogramPeak:

    ChromatogramPeak()
    ChromatogramPeak(ChromatogramPeak &)
    bool operator==(ChromatogramPeak)
    bool operator!=(ChromatogramPeak)

    IntensityType getIntensity()
```

```
    void setIntensity(IntensityType)

    DPosition1 getPosition()
    void setPosition(DPosition1)

    CoordinateType getRT()
    void setRT(CoordinateType)

    CoordinateType getPos()
    void setPos(CoordinateType)

    CoordinateType getMZ()
    void setMZ(CoordinateType)
```

6.33 ChromatogramSettings

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/ChromatogramSettings.h>" namespace "OpenMS":

  cdef cppclass ChromatogramSettings:

    ChromatogramSettings()
    ChromatogramSettings(ChromatogramSettings)

    Product getProduct()
    void setProduct(Product p)

    String getNativeID()
    void setNativeID(String native_id)

    String getComment()
    void setComment(String comment)

    InstrumentSettings getInstrumentSettings()
    void setInstrumentSettings(InstrumentSettings instrument_settings)

    AcquisitionInfo getAcquisitionInfo()
    void setAcquisitionInfo(AcquisitionInfo acquisition_info)

    SourceFile getSourceFile()
    void setSourceFile(SourceFile source_file)

    Precursor getPrecursor()
    void setPrecursor(Precursor precursor)
```

```
    libcpp_vector[ shared_ptr[DataProcessing] ] getDataProcessing()
    void setDataProcessing(libcpp_vector[ shared_ptr[DataProcessing] ])

    void setChromatogramType(ChromatogramType type)
    ChromatogramType getChromatogramType()

cdef extern from "<OpenMS/METADATA/ChromatogramSettings.h>" namespace\
       "OpenMS::ChromatogramSettings":

  cdef enum ChromatogramType:
    # wrap-attach:
    #   ChromatogramSettings

    MASS_CHROMATOGRAM,
    TOTAL_ION_CURRENT_CHROMATOGRAM,
    SELECTED_ION_CURRENT_CHROMATOGRAM,
    BASEPEAK_CHROMATOGRAM,
    SELECTED_ION_MONITORING_CHROMATOGRAM,
    SELECTED_REACTION_MONITORING_CHROMATOGRAM,
    ELECTROMAGNETIC_RADIATION_CHROMATOGRAM,
    ABSORPTION_CHROMATOGRAM,
    EMISSION_CHROMATOGRAM,
    SIZE_OF_CHROMATOGRAM_TYPE
```

6.34 ChromatogramTools

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/KERNEL/ChromatogramTools.h>" namespace "OpenMS":

  cdef cppclass ChromatogramTools:
    ChromatogramTools()

    void convertChromatogramsToSpectra(
        MSExperiment[Peak1D, ChromatogramPeak] & epx
        )

    void convertSpectraToChromatograms(
        MSExperiment[Peak1D, ChromatogramPeak] & epx,
        int remove_spectra
        )
```

6.35 ClusteringGrid

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/COMPARISON/CLUSTERING/ClusteringGrid.h>" namespace\
       "OpenMS":

  cdef cppclass ClusteringGrid "OpenMS::ClusteringGrid":
    ClusteringGrid(ClusteringGrid)  #wrap-ignore
    ClusteringGrid(libcpp_vector[ double ] & grid_spacing_x, libcpp_vector[ double ]\
       & grid_spacing_y)
    libcpp_vector[ double ] getGridSpacingX()
    libcpp_vector[ double ] getGridSpacingY()
    void addCluster(libcpp_pair[int,int] cell_index, int & cluster_index)
    void removeCluster(libcpp_pair[int,int] cell_index, int & cluster_index)
    void removeAllClusters()
    # NAMESPACE # std::list[ int ] getClusters(CellIndex & cell_index)
    libcpp_pair[int,int] getIndex(DPosition2 position)
    bool isNonEmptyCell(libcpp_pair[int,int] cell_index)
    int getCellCount()
```

6.36 CompNovoIdentification

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/DENOVO/CompNovoIdentification.h>" namespace\
       "OpenMS":

  cdef cppclass CompNovoIdentification(DefaultParamHandler):
    # wrap-inherits:
    #  DefaultParamHandler

    CompNovoIdentification()
    CompNovoIdentification(CompNovoIdentification)     #private

    void getIdentifications(libcpp_vector[PeptideIdentification] & ids,\
       MSExperiment[Peak1D,ChromatogramPeak])
    void getIdentification(PeptideIdentification & id, MSSpectrum[Peak1D] cid_spec,\
       MSSpectrum[Peak1D] etd_spec)
```

6.37 CompNovoIdentificationCID

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/DENOVO/CompNovoIdentificationCID.h>" namespace\
       "OpenMS":

  cdef cppclass CompNovoIdentificationCID(DefaultParamHandler):
    # wrap-inherits:
    #  DefaultParamHandler

    CompNovoIdentificationCID()
    CompNovoIdentificationCID(CompNovoIdentificationCID)     #private

    void getIdentifications(libcpp_vector[PeptideIdentification] & ids,\
       MSExperiment[Peak1D,ChromatogramPeak])
    void getIdentification(PeptideIdentification & id, MSSpectrum[Peak1D] cid_spec)
```

6.38 CompNovoIonScoring

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/DENOVO/CompNovoIonScoring.h>" namespace\
       "OpenMS":

  cdef cppclass CompNovoIonScoring "OpenMS::CompNovoIonScoring":
    CompNovoIonScoring()
    CompNovoIonScoring(CompNovoIonScoring)
```

6.39 CompNovoIonScoringCID

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/DENOVO/CompNovoIonScoringCID.h>" namespace\
       "OpenMS":

  cdef cppclass CompNovoIonScoringCID "OpenMS::CompNovoIonScoringCID":
    CompNovoIonScoringCID()
    CompNovoIonScoringCID(CompNovoIonScoringCID)
```

6.40 ComplementFilter

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/TRANSFORMERS/ComplementFilter.h>" namespace\
       "OpenMS":

  cdef cppclass ComplementFilter(FilterFunctor) :
    # wrap-inherits:
    #  FilterFunctor
    ComplementFilter()
    ComplementFilter(ComplementFilter)
    double apply(MSSpectrum[Peak1D] & )
    # POINTER # FilterFunctor * create()
    String getProductName()
```

6.41 ComplementMarker

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/TRANSFORMERS/ComplementMarker.h>" namespace\
       "OpenMS":

  cdef cppclass ComplementMarker(PeakMarker) :
    # wrap-inherits:
    #  PeakMarker
    ComplementMarker()
    ComplementMarker(ComplementMarker)
    void apply(libcpp_map[ double, bool ] & , MSSpectrum[Peak1D] & )
    PeakMarker * create()  # wrap-ignore
    # String getProductName()
```

6.42 Compomer

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/DATASTRUCTURES/Compomer.h>" namespace "OpenMS":

  cdef cppclass Compomer:

    Compomer()
    Compomer(Compomer)

    void add(Adduct & a, UInt side)

    bool isConflicting(Compomer & cmp, UInt side_this, UInt side_other)
```

```
    void setID(Size id)

    Size getID()

    libcpp_vector[Map[String, Adduct] ] getComponent()  # wrap-ignore

    Int getNetCharge()

    double getMass()

    Int getPositiveCharges()

    Int getNegativeCharges()

    double getLogP()

    double getRTShift()

    String getAdductsAsString()

    String getAdductsAsString(UInt side)

    bool isSingleAdduct(Adduct & a, UInt side)

    Compomer removeAdduct(Adduct & a)

    Compomer removeAdduct(Adduct & a, UInt side)

    StringList getLabels(UInt side)

cdef extern from "<OpenMS/DATASTRUCTURES/Compomer.h>" namespace "OpenMS::Compomer":
  cdef enum SIDE: LEFT, RIGHT, BOTH

cdef extern from "<OpenMS/DATASTRUCTURES/Compomer.h>" namespace "OpenMS":

  cdef cppclass Compomer:

    Compomer()
    Compomer(Compomer)

    void add(Adduct & a, UInt side)

    bool isConflicting(Compomer & cmp, UInt side_this, UInt side_other)

    void setID(Size id)

    Size getID()
```

```
    libcpp_vector[Map[String, Adduct] ] getComponent()  # wrap-ignore

    Int getNetCharge()

    double getMass()

    Int getPositiveCharges()

    Int getNegativeCharges()

    double getLogP()

    double getRTShift()

    String getAdductsAsString()

    String getAdductsAsString(UInt side)

    bool isSingleAdduct(Adduct & a, UInt side)

    Compomer removeAdduct(Adduct & a)

    Compomer removeAdduct(Adduct & a, UInt side)

    StringList getLabels(UInt side)

cdef extern from "<OpenMS/DATASTRUCTURES/Compomer.h>" namespace "OpenMS::Compomer":
  cdef enum SIDE: LEFT, RIGHT, BOTH
```

6.43 ConfidenceScoring

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/OPENSWATH/ConfidenceScoring.h>" namespace\
       "OpenMS":

  cdef cppclass ConfidenceScoring:

    ConfidenceScoring()
    ConfidenceScoring(ConfidenceScoring)

    void initialize(TargetedExperiment & targeted, Size n_decoys, Size\
       n_transitions, TransformationDescription trafo)
    void initializeGlm(double intercept, double rt_coef, double int_coef)
    void scoreMap(FeatureMap & map)
```

6.44 ConsensusFeature

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/KERNEL/ConsensusFeature.h>" namespace "OpenMS":

  cdef cppclass ConsensusFeature(UniqueIdInterface,Peak2D):
    # wrap-inherits:
    #  UniqueIdInterface
    #  Peak2D

    ConsensusFeature()
    ConsensusFeature(ConsensusFeature &)
    ConsensusFeature(UInt64, Peak2D, UInt64)
    ConsensusFeature(UInt64, BaseFeature)
    ConsensusFeature(UInt64, ConsensusFeature)

    void computeConsensus()
    void computeMonoisotopicConsensus()
    void computeDechargeConsensus(FeatureMap, bool)

    void insert(UInt64, Peak2D, UInt64)
    void insert(UInt64, BaseFeature)
    void insert(UInt64, ConsensusFeature)

    float getQuality()
    void setQuality(float q)

    float getWidth()
    void setWidth(float q)

    Int getCharge()
    void setCharge(Int q)

    libcpp_vector[FeatureHandle] getFeatureList()

    Size size()

    libcpp_vector[PeptideIdentification] getPeptideIdentifications()
    void setPeptideIdentifications(libcpp_vector[PeptideIdentification] & peptides)

    bool operator==(ConsensusFeature)
    bool operator!=(ConsensusFeature)
```

```
    void getKeys(libcpp_vector[String] & keys)
    void getKeys(libcpp_vector[unsigned int] & keys)  # wrap-as:getKeysAsIntegers
    DataValue getMetaValue(unsigned int)
    DataValue getMetaValue(String)
    void setMetaValue(unsigned int, DataValue)
    void setMetaValue(String, DataValue)
    bool metaValueExists(String)
    bool metaValueExists(unsigned int)
    void removeMetaValue(String)
    void removeMetaValue(unsigned int)
    void clearMetaInfo()

    void addRatio(Ratio r)
    void setRatios(libcpp_vector[Ratio] rs)
    libcpp_vector[Ratio] getRatios()

    void clear()
    bool empty()

cdef extern from "<OpenMS/KERNEL/ConsensusFeature.h>" namespace\
       "OpenMS::ConsensusFeature":

  cdef cppclass Ratio:

    Ratio()
    Ratio(Ratio rhs)

    double ratio_value_
    String denominator_ref_
    String numerator_ref_
    libcpp_vector[String] description_
```

6.45 ConsensusIDAlgorithm

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/ID/ConsensusIDAlgorithm.h>" namespace "OpenMS":

  cdef cppclass ConsensusIDAlgorithm(DefaultParamHandler) :
    # wrap-inherits:
    #  DefaultParamHandler
    # wrap-ignore
    # ABSTRACT class
    ConsensusIDAlgorithm()  #wrap-ignore
    ConsensusIDAlgorithm(ConsensusIDAlgorithm)  #wrap-ignore
```

```
    void apply(libcpp_vector[ PeptideIdentification ] & ids, Size number_of_runs)
```

6.46 ConsensusIDAlgorithmAverage

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/ID/ConsensusIDAlgorithmAverage.h>" namespace\
       "OpenMS":

  cdef cppclass ConsensusIDAlgorithmAverage(ConsensusIDAlgorithmIdentity) :
    # wrap-inherits:
    #  ConsensusIDAlgorithmIdentity
    ConsensusIDAlgorithmAverage()
    ConsensusIDAlgorithmAverage(ConsensusIDAlgorithmAverage)  #wrap-ignore
```

6.47 ConsensusIDAlgorithmBest

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/ID/ConsensusIDAlgorithmBest.h>" namespace\
       "OpenMS":

  cdef cppclass ConsensusIDAlgorithmBest(ConsensusIDAlgorithmIdentity) :
    # wrap-inherits:
    #  ConsensusIDAlgorithmIdentity
    ConsensusIDAlgorithmBest()
    ConsensusIDAlgorithmBest(ConsensusIDAlgorithmBest)  #wrap-ignore
```

6.48 ConsensusIDAlgorithmIdentity

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/ID/ConsensusIDAlgorithmIdentity.h>" namespace\
       "OpenMS":

  cdef cppclass ConsensusIDAlgorithmIdentity(ConsensusIDAlgorithm) :
    # wrap-inherits:
    #  ConsensusIDAlgorithm
    # wrap-ignore
```

```
    # ABSTRACT class
    ConsensusIDAlgorithmIdentity()  #wrap-ignore
    ConsensusIDAlgorithmIdentity(ConsensusIDAlgorithmIdentity)  #wrap-ignore
```

6.49 ConsensusIDAlgorithmPEPIons

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/ID/ConsensusIDAlgorithmPEPIons.h>" namespace\
       "OpenMS":

  cdef cppclass ConsensusIDAlgorithmPEPIons(ConsensusIDAlgorithmSimilarity) :
    # wrap-inherits:
    #  ConsensusIDAlgorithmSimilarity
    ConsensusIDAlgorithmPEPIons()
    ConsensusIDAlgorithmPEPIons(ConsensusIDAlgorithmPEPIons)  #wrap-ignore
```

6.50 ConsensusIDAlgorithmPEPMatrix

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/ID/ConsensusIDAlgorithmPEPMatrix.h>" namespace\
       "OpenMS":

  cdef cppclass ConsensusIDAlgorithmPEPMatrix(ConsensusIDAlgorithmSimilarity) :
    # wrap-inherits:
    #  ConsensusIDAlgorithmSimilarity
    ConsensusIDAlgorithmPEPMatrix()
    ConsensusIDAlgorithmPEPMatrix(ConsensusIDAlgorithmPEPMatrix)  #wrap-ignore
```

6.51 ConsensusIDAlgorithmRanks

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/ID/ConsensusIDAlgorithmRanks.h>" namespace\
       "OpenMS":

  cdef cppclass ConsensusIDAlgorithmRanks(ConsensusIDAlgorithmIdentity) :
    # wrap-inherits:
```

```
    #  ConsensusIDAlgorithmIdentity
    ConsensusIDAlgorithmRanks()
    ConsensusIDAlgorithmRanks(ConsensusIDAlgorithmRanks)  #wrap-ignore
```

6.52 ConsensusIDAlgorithmSimilarity

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/ID/ConsensusIDAlgorithmSimilarity.h>" namespace\
       "OpenMS":

  cdef cppclass ConsensusIDAlgorithmSimilarity(ConsensusIDAlgorithm) :
    # wrap-inherits:
    #  ConsensusIDAlgorithm
    # wrap-ignore
    # ABSTRACT class
    ConsensusIDAlgorithmSimilarity()  #wrap-ignore
    ConsensusIDAlgorithmSimilarity(ConsensusIDAlgorithmSimilarity)  #wrap-ignore
```

6.53 ConsensusIDAlgorithmWorst

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/ID/ConsensusIDAlgorithmWorst.h>" namespace\
       "OpenMS":

  cdef cppclass ConsensusIDAlgorithmWorst(ConsensusIDAlgorithmIdentity) :
    # wrap-inherits:
    #  ConsensusIDAlgorithmIdentity
    ConsensusIDAlgorithmWorst()
    ConsensusIDAlgorithmWorst(ConsensusIDAlgorithmWorst)  #wrap-ignore
```

6.54 ConsensusMap

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/KERNEL/ConsensusMap.h>" namespace "OpenMS::ConsensusMap":

  cdef cppclass FileDescription:
```

```
    String filename
    String label
    Size size
    UInt64 unique_id

    FileDescription()
    FileDescription(FileDescription &)

  ctypedef libcpp_map[unsigned long int, FileDescription] FileDescriptions\
      "OpenMS::ConsensusMap::FileDescriptions"
  ctypedef libcpp_map[unsigned long int, FileDescription].iterator\
      FileDescriptions_iterator "OpenMS::ConsensusMap::FileDescriptions::iterator"

cdef extern from "<OpenMS/KERNEL/ConsensusMap.h>" namespace "OpenMS":

  cdef cppclass ConsensusMap(UniqueIdInterface, DocumentIdentifier, RangeManager2):

    # wrap-inherits:
    #   UniqueIdInterface
    #   DocumentIdentifier
    #   RangeManager2

    ConsensusMap()
    ConsensusMap(ConsensusMap &)

    bool operator==(ConsensusMap)
    bool operator!=(ConsensusMap)

    int size()
    bool empty()
    void reserve(Size s)
    ConsensusFeature operator[](int)  #wrap-upper-limit:size()
    void push_back(ConsensusFeature spec)

    ConsensusMap iadd(ConsensusMap)  # wrap-as:operator+=

    void clear(bool clear_meta_data)
    void clear()

    void updateRanges()

    libcpp_vector[ProteinIdentification] getProteinIdentifications(
        ) nogil except+

    void setProteinIdentifications(
        libcpp_vector[ProteinIdentification]
        ) nogil except+
```

```
    libcpp_vector[PeptideIdentification]\
        getUnassignedPeptideIdentifications() nogil except+

    void setUnassignedPeptideIdentifications(
        libcpp_vector[PeptideIdentification]
        ) nogil except+

    libcpp_vector[DataProcessing] getDataProcessing()
    void setDataProcessing(libcpp_vector[DataProcessing])

    void setPrimaryMSRunPath(StringList& s)
    StringList getPrimaryMSRunPath()

    libcpp_vector[ConsensusFeature].iterator begin(
        )    # wrap-iter-begin:__iter__(ConsensusFeature)
    libcpp_vector[ConsensusFeature].iterator end(
        )     # wrap-iter-end:__iter__(ConsensusFeature)

    void applyMemberFunction(Size(* fun)())  # wrap-ignore

    void sortByIntensity(bool reverse)
    void sortByIntensity()
    void sortByRT()
    void sortByMZ()
    void sortByPosition()
    void sortByQuality(bool reverse)
    void sortByQuality()
    void sortBySize()
    void sortByMaps()

    void setFileDescriptions(FileDescriptions &)   #wrap-ignore
    FileDescriptions & getFileDescriptions()     #wrap-ignore

    String getExperimentType()
    void setExperimentType(String experiment_type)
```

6.55 ConsensusMapNormalizerAlgorithmMedian

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from\
       "<OpenMS/ANALYSIS/MAPMATCHING/ConsensusMapNormalizerAlgorithmMedian.h>" namespace\
       "OpenMS::ConsensusMapNormalizerAlgorithmMedian":

  cdef enum NormalizationMethod:
```

```
    NM_SCALE,
    NM_SHIFT

cdef extern from\
       "<OpenMS/ANALYSIS/MAPMATCHING/ConsensusMapNormalizerAlgorithmMedian.h>" namespace\
       "OpenMS":
  cdef cppclass ConsensusMapNormalizerAlgorithmMedian:

    ConsensusMapNormalizerAlgorithmMedian()
    ConsensusMapNormalizerAlgorithmMedian(ConsensusMapNormalizerAlgorithmMedian) \
       #wrap-ignore

    Size computeMedians(ConsensusMap & input_map, libcpp_vector[double] & medians,\
       String & acc_filter, String & desc_filter)
    void normalizeMaps(ConsensusMap & input_map, NormalizationMethod method, String\
       & acc_filter, String & desc_filter)
```

6.56 ConsensusMapNormalizerAlgorithmQuantile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from\
       "<OpenMS/ANALYSIS/MAPMATCHING/ConsensusMapNormalizerAlgorithmQuantile.h>" namespace\
       "OpenMS":

  cdef cppclass ConsensusMapNormalizerAlgorithmQuantile:

    ConsensusMapNormalizerAlgorithmQuantile()
    ConsensusMapNormalizerAlgorithmQuantile(ConsensusMapNormalizerAlgorithmQuantile)\
        #wrap-ignore

    void normalizeMaps(ConsensusMap & input_map)

    void resample(libcpp_vector[ double ] & data_in, libcpp_vector[ double ] &\
       data_out, UInt n_resampling_points)
    void extractIntensityVectors(ConsensusMap & map_, libcpp_vector[ libcpp_vector[\
       double ] ] & out_intensities)
    void setNormalizedIntensityValues(libcpp_vector[ libcpp_vector[ double ] ] &\
       feature_ints, ConsensusMap & map_)
```

6.57 ConsensusMapNormalizerAlgorithmThreshold

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from\
       "<OpenMS/ANALYSIS/MAPMATCHING/ConsensusMapNormalizerAlgorithmThreshold.h>"\
       namespace "OpenMS":

  cdef cppclass ConsensusMapNormalizerAlgorithmThreshold:

    ConsensusMapNormalizerAlgorithmThreshold()
   \
       ConsensusMapNormalizerAlgorithmThreshold(ConsensusMapNormalizerAlgorithmThreshold) \
       #wrap-ignore

    libcpp_vector[double] computeCorrelation(ConsensusMap & input_map, double\
       ratio_threshold, String & acc_filter, String & desc_filter)
    void normalizeMaps(ConsensusMap & input_map, libcpp_vector[double] & ratios)
```

6.58 ConsensusXMLFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/ConsensusXMLFile.h>" namespace "OpenMS":

  cdef cppclass ConsensusXMLFile:
    ConsensusXMLFile()

    void load(String, ConsensusMap &) nogil except+
    void store(String, ConsensusMap &) nogil except+

    PeakFileOptions getOptions()
```

6.59 ContactPerson

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/ContactPerson.h>" namespace "OpenMS":

  cdef cppclass ContactPerson(MetaInfoInterface):
    # wrap-inherits:
    #  MetaInfoInterface

    ContactPerson()
    ContactPerson(ContactPerson)  # wrap-ignore

    String getFirstName()
```

```
    void setFirstName(String name)

    String getLastName()
    void setLastName(String name)

    void setName(String name)

    String getInstitution()
    void setInstitution(String institution)

    String getEmail()
    void setEmail(String email)

    String getURL()
    void setURL(String email)

    String getAddress()
    void setAddress(String email)

    String getContactInfo()
    void setContactInfo(String contact_info)
```

6.60 ContinuousWaveletTransform

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/TRANSFORMATIONS/RAW2PEAK/ContinuousWaveletTransform.h>"\
       namespace "OpenMS":

  cdef cppclass ContinuousWaveletTransform "OpenMS::ContinuousWaveletTransform":
    ContinuousWaveletTransform()
    ContinuousWaveletTransform(ContinuousWaveletTransform)  #wrap-ignore
    libcpp_vector[ Peak1D ]  getSignal()
    void setSignal(libcpp_vector[ Peak1D ] & signal)
    libcpp_vector[ double ]  getWavelet()
    void setWavelet(libcpp_vector[ double ] & wavelet)
    double  getScale()
    void setScale(double scale)
    double  getSpacing()
    void setSpacing(double spacing)
    SignedSize getLeftPaddingIndex()
    void setLeftPaddingIndex(SignedSize end_left_padding)
    SignedSize getRightPaddingIndex()
    void setRightPaddingIndex(SignedSize begin_right_padding)
    SignedSize getSignalLength()
```

```
    void setSignalLength(SignedSize signal_length)
    int getSize()
    void init(double scale, double spacing)
    # double <](unsigned int i)
    # double <](unsigned int i)
```

6.61 ContinuousWaveletTransformNumIntegration

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from\
       "<OpenMS/TRANSFORMATIONS/RAW2PEAK/ContinuousWaveletTransformNumIntegration.h>"\
       namespace "OpenMS":

  cdef cppclass ContinuousWaveletTransformNumIntegration(ContinuousWaveletTransform) :
    # wrap-inherits:
    #  ContinuousWaveletTransform
    ContinuousWaveletTransformNumIntegration()
   \
       ContinuousWaveletTransformNumIntegration(ContinuousWaveletTransformNumIntegration) \
       #wrap-ignore
    # TODO iterator
    # TEMPLATE # void transform(InputPeakIterator begin_input, InputPeakIterator\
       end_input, float resolution, unsigned int zeros)
```

6.62 ControlledVocabulary

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/ControlledVocabulary.h>" namespace "OpenMS":

  cdef cppclass ControlledVocabulary:

    ControlledVocabulary()
    ControlledVocabulary(ControlledVocabulary & voc)

    String name()

    void loadFromOBO(String name, String filename)

    bool exists(String id)
```

```
    bool hasTermWithName(String name)

    CVTerm_ControlledVocabulary getTerm(String id)

    CVTerm_ControlledVocabulary getTermByName(String name, String desc)

    void getAllChildTerms(libcpp_set[String] terms, String parent)

    bool isChildOf(String child, String parent)

cdef extern from "<OpenMS/FORMAT/ControlledVocabulary.h>" namespace\
       "OpenMS::ControlledVocabulary":

  cdef cppclass CVTerm_ControlledVocabulary "OpenMS::ControlledVocabulary::CVTerm":

    String name  #< Text name
    String id  #< Identifier
    libcpp_set[String] parents  #< The parent IDs
    libcpp_set[String] children  #< The child IDs
    bool obsolete  #< Flag that indicates of the term is obsolete
    String description  #< Term description
    StringList synonyms  #< List of synonyms
    StringList unparsed  #< Unparsed lines from the definition file
    XRefType_CVTerm_ControlledVocabulary xref_type  #< xref value-type for the\
       CV-term
    StringList xref_binary  #< xref binary-data-type for the CV-term (list of all\
       allowed data value types for the current binary data array)
    libcpp_set[String] units  #< unit accession ids, defined by relationship has units

    CVTerm_ControlledVocabulary()
    CVTerm_ControlledVocabulary(CVTerm_ControlledVocabulary rhs)

    String toXMLString(String ref, String value)
    String toXMLString(String ref, DataValue value)
    String getXRefTypeName(XRefType_CVTerm_ControlledVocabulary type)
    bool isHigherBetterScore(CVTerm_ControlledVocabulary term)

cdef extern from "<OpenMS/FORMAT/ControlledVocabulary.h>" namespace\
       "OpenMS::ControlledVocabulary::CVTerm":

  cdef enum XRefType_CVTerm_ControlledVocabulary\
       "OpenMS::ControlledVocabulary::CVTerm::XRefType":
    XSD_STRING = 0, # xsd:string A string
    XSD_INTEGER, # xsd:integer Any integer
    XSD_DECIMAL, # xsd:decimal Any real number
    XSD_NEGATIVE_INTEGER, # xsd:negativeInteger Any negative integer
    XSD_POSITIVE_INTEGER, # xsd:positiveInteger Any integer ] 0
    XSD_NON_NEGATIVE_INTEGER, # xsd:nonNegativeInteger Any integer ]= 0
```

```
    XSD_NON_POSITIVE_INTEGER, # xsd:nonPositiveInteger Any integer < 0
    XSD_BOOLEAN, # xsd:boolean True or false
    XSD_DATE, # xsd:date An XML-Schema date
    XSD_ANYURI, # xsd:anyURI uniform resource identifier
    NONE
```

6.63 ConversionHelper

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/KERNEL/ConversionHelper.h>" namespace "OpenMS":

  cdef cppclass MapConversion:

    void convert(UInt64 input_map_index,
           FeatureMap input_map,
           ConsensusMap & output_map,
           Size n)

    void convert(UInt64 input_map_index,
           MSExperiment[Peak1D, ChromatogramPeak] & input_map,
           ConsensusMap & output_map,
           Size n)

    void convert(ConsensusMap input_map,
           bool keep_uids,
           FeatureMap & output_map)
```

6.64 ConvexHull2D

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/DATASTRUCTURES/ConvexHull2D.h>" namespace "OpenMS":

  cdef cppclass ConvexHull2D:

    ConvexHull2D()
    ConvexHull2D(ConvexHull2D)  # wrap-ignore

    bool operator==(ConvexHull2D)
    void clear()
    Size compress()
```

```
    void expandToBoundingBox()

    bool addPoint(DPosition2 point)
    void addPoints(libcpp_vector[DPosition2] points)

    bool encloses(DPosition2)
    libcpp_vector[DPosition2] getHullPoints()
    void setHullPoints(libcpp_vector[DPosition2] )
    DBoundingBox2 getBoundingBox()
```

6.65 CubicSpline2d

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/MATH/MISC/CubicSpline2d.h>" namespace "OpenMS":

  cdef cppclass CubicSpline2d "OpenMS::CubicSpline2d":

    CubicSpline2d(CubicSpline2d)  #wrap-ignore
    CubicSpline2d(libcpp_vector[ double ] x, libcpp_vector[ double ] y)
    CubicSpline2d(libcpp_map[ double, double ] m)

    double eval(double x)
    double derivatives(double x, unsigned order)
```

6.66 DBoundingBox

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/DATASTRUCTURES/DBoundingBox.h>" namespace "OpenMS":

  cdef cppclass DBoundingBox2 "OpenMS::DBoundingBox<2> ":
    DBoundingBox2()
    DBoundingBox2(DBoundingBox2)
    DPosition2 minPosition()
    DPosition2 maxPosition()
```

6.67 DIAScoring

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/OPENSWATH/DIAScoring.h>" namespace "OpenMS":

  cdef cppclass DIAScoring:
    DIAScoring()
    void set_dia_parameters(double dia_extract_window, double dia_centroided,
                double dia_byseries_intensity_min, double
                dia_byseries_ppm_diff, double dia_nr_isotopes,
                double dia_nr_charges)

    bool dia_ms1_massdiff_score(double precursor_mz, OSSpectrumPtr spectrum,
                  double& ppm_score)

    void dia_ms1_isotope_scores(double precursor_mz, OSSpectrumPtr spectrum, size_t\
       charge_state,
                  double& isotope_corr, double& isotope_overlap, libcpp_string\
       sum_formula)

    void dia_by_ion_score(OSSpectrumPtr spectrum, AASequence sequence, int charge,\
       double & bseries_score, double & yseries_score)  #\
       wrap-return:return(bseries_score,yseries_score) wrap-ignore

    void score_with_isotopes(OSSpectrumPtr spectrum, libcpp_vector[LightTransition]\
       transitions,
                 double& dotprod, double& manhattan)
```

6.68 DPosition

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/DATASTRUCTURES/DPosition.h>" namespace "OpenMS":

  cdef cppclass DPosition1 "OpenMS::DPosition<1> ":
    # wrap-ignore
    DPosition1()
    DPosition1(DPosition1)

    double & operator[](Size index)
    bool operator==(DPosition1)
    bool operator!=(DPosition1)

    bool operator<(DPosition1)
    bool operator<=(DPosition1)

    bool operator>(DPosition1)
```

```
    bool operator>=(DPosition1)

    void clear()
    Size size()

  cdef cppclass DPosition2 "OpenMS::DPosition<2> ":
    # wrap-ignore
    DPosition2()
    DPosition2(DPosition2)

    double & operator[](Size index)
    bool operator==(DPosition2)
    bool operator!=(DPosition2)

    bool operator<(DPosition2)
    bool operator<=(DPosition2)

    bool operator>(DPosition2)
    bool operator>=(DPosition2)

    void clear()
    Size size()
```

6.69 DRange

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/DATASTRUCTURES/DRange.h>" namespace "OpenMS":

  cdef cppclass DRange1 "OpenMS::DRange<1> ":
    DRange1()
    DRange1(DRange1)
    DRange1(DPosition1 lower, DPosition1 upper)
    bool operator==(DRange1 & rhs)
    bool encloses(DPosition1 & position)
    DRange1 united(DRange1 other_range)
    bool isIntersected(DRange1 & range_)
    bool isEmpty()

  cdef cppclass DRange2 "OpenMS::DRange<2> ":
    DRange2()
    DRange2(DRange2)
    DRange2(DPosition2 lower, DPosition2 upper)
    DRange2(double minx, double miny, double maxx, double maxy)
    bool operator==(DRange2 & rhs)
    DRange2 united(DRange2 other_range)
```

```
    bool isIntersected(DRange2 & range_)
    bool isEmpty()

cdef extern from "<OpenMS/DATASTRUCTURES/DRange.h>" namespace "OpenMS::DRange":
  cdef enum DRangeIntersection "OpenMS::DRange::DRangeIntersection":
    Disjoint
    Intersects
    Inside
```

6.70 DTA2DFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/DTA2DFile.h>" namespace "OpenMS":

  cdef cppclass DTA2DFile(ProgressLogger):
    # wrap-inherits:
    #  ProgressLogger

    DTA2DFile()
    DTA2DFile(DTA2DFile)

    void storeTIC(String filename, MSExperiment[Peak1D,ChromatogramPeak] & peakmap)
    void store(String filename, MSExperiment[Peak1D,ChromatogramPeak] & peakmap)
    void load(String filename, MSExperiment[Peak1D,ChromatogramPeak] & peakmap)
    PeakFileOptions  getOptions()
```

6.71 DTAFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/DTAFile.h>" namespace "OpenMS":

  cdef cppclass DTAFile:

    DTAFile()
    DTAFile(DTAFile)

    void load(String filename, MSSpectrum[Peak1D] & spectrum)
    void store(String filename, MSSpectrum[Peak1D] & spectrum)
```

6.72 DataFilters

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/DATAREDUCTION/DataFilters.h>" namespace\
       "OpenMS":

  cdef cppclass DataFilters "OpenMS::DataFilters":
    DataFilters()
    DataFilters(DataFilters)  #wrap-ignore
    Size size()
    DataFilter operator[](size_t)  # wrap-upper-limit:size()
    void add(DataFilter & filter_)
    void remove(Size index)
    void replace(Size index, DataFilter & filter_)
    void clear()
    void setActive(bool is_active)
    bool isActive()
    bool passes(Feature & feature)
    bool passes(ConsensusFeature & consensus_feature)
    bool passes(MSSpectrum[ Peak1D ] & spectrum, Size peak_index)

cdef extern from "<OpenMS/FILTERING/DATAREDUCTION/DataFilters.h>" namespace\
       "OpenMS::DataFilters":

  cdef cppclass DataFilter "OpenMS::DataFilters::DataFilter":
    DataFilter()
    DataFilter(DataFilter)  #wrap-ignore
    FilterType field
    FilterOperation op
    double value
    String value_string
    String meta_name
    bool value_is_numerical
    String toString()
    void fromString(String & filter_)
    bool operator==(DataFilter & rhs)
    bool operator!=(DataFilter & rhs)

cdef extern from "<OpenMS/FILTERING/DATAREDUCTION/DataFilters.h>" namespace\
       "OpenMS::DataFilters":
  cdef enum FilterType "OpenMS::DataFilters::FilterType":
    #wrap-attach:
    #  DataFilters
    INTENSITY
    QUALITY
    CHARGE
```

```
    SIZE
    META_DATA

cdef extern from "<OpenMS/FILTERING/DATAREDUCTION/DataFilters.h>" namespace\
       "OpenMS::DataFilters":
  cdef enum FilterOperation "OpenMS::DataFilters::FilterOperation":
    #wrap-attach:
    #  DataFilters
    GREATER_EQUAL
    EQUAL
    LESS_EQUAL
    EXISTS
```

6.73 DataProcessing

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/DataProcessing.h>" namespace "OpenMS":

  cdef cppclass DataProcessing(MetaInfoInterface):
    # wrap-inherits:
    #   MetaInfoInterface

    DataProcessing()
    DataProcessing(DataProcessing)  # wrap-ignore

    void setProcessingActions(libcpp_set[ProcessingAction])
    libcpp_set[ProcessingAction] getProcessingActions()

    Software getSoftware()
    void setSoftware(Software s)

    DateTime getCompletionTime()
    void setCompletionTime(DateTime t)

    void getKeys(libcpp_vector[String] & keys)
    void getKeys(libcpp_vector[unsigned int] & keys)  # wrap-as:getKeysAsIntegers
    DataValue getMetaValue(unsigned int)
    DataValue getMetaValue(String)
    void setMetaValue(unsigned int, DataValue)
    void setMetaValue(String, DataValue)
    bool metaValueExists(String)
    bool metaValueExists(unsigned int)
    void removeMetaValue(String)
    void removeMetaValue(unsigned int)
```

```
  ctypedef shared_ptr[DataProcessing] DataProcessingPtr

cdef extern from "<OpenMS/METADATA/DataProcessing.h>" namespace\
       "OpenMS::DataProcessing":

  cdef enum ProcessingAction:

    DATA_PROCESSING, CHARGE_DECONVOLUTION, DEISOTOPING, SMOOTHING,
    CHARGE_CALCULATION, PRECURSOR_RECALCULATION, BASELINE_REDUCTION,
    PEAK_PICKING, ALIGNMENT, CALIBRATION, NORMALIZATION, FILTERING,
    QUANTITATION, FEATURE_GROUPING, IDENTIFICATION_MAPPING,
    FORMAT_CONVERSION, CONVERSION_MZDATA, CONVERSION_MZML,
    CONVERSION_MZXML, CONVERSION_DTA, SIZE_OF_PROCESSINGACTION
```

6.74 DataValue

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/DATASTRUCTURES/DataValue.h>" namespace "OpenMS":

  cdef cppclass DataValue:
     DataValue()
     DataValue(DataValue)  # wrap-ignore
     DataValue(char *)
     DataValue(int)
     DataValue(double)
     DataValue(StringList)
     DataValue(IntList)
     DataValue(DoubleList)

     int operator()(int)    #wrap-cast:toInt
     libcpp_string operator()(DataValue)    #wrap-cast:toString
     double operator()(DataValue)       #wrap-cast:toDouble
     StringList toStringList()
     libcpp_vector[ double ] toDoubleList()
     libcpp_vector[ int ] toIntList()

     DataType valueType()
     int isEmpty()

     String toString()
     bool toBool()
     bool hasUnit()
     String  getUnit()
```

```
    void setUnit(String & unit)

cdef extern from "<OpenMS/DATASTRUCTURES/DataValue.h>" namespace\
      "OpenMS::DataValue":

  cdef enum DataType:
     STRING_VALUE, INT_VALUE, DOUBLE_VALUE, STRING_LIST, INT_LIST, \
     DOUBLE_LIST, EMPTY_VALUE
```

6.75 Date

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/DATASTRUCTURES/Date.h>" namespace "OpenMS":

  cdef cppclass Date:

    Date()
    Date(Date)  # wrap-ignore

    void set(String & date)
    Date today()
    String get()
    void clear()
```

6.76 DateTime

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/DATASTRUCTURES/DateTime.h>" namespace "OpenMS":

  cdef cppclass DateTime:

    DateTime()
    DateTime(DateTime)  # wrap-ignore

    void setDate(String date)

    void setTime(String date)

    String getDate()

    String getTime()
```

```
    DateTime now()

    void clear()

    String get()

    void set(String date)

cdef extern from "<OpenMS/DATASTRUCTURES/DateTime.h>" namespace "OpenMS::DateTime":

  DateTime now() # wrap-attach:DateTime
```

6.77 DeNovoAlgorithm

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/DENOVO/DeNovoAlgorithm.h>" namespace "OpenMS":

  cdef cppclass DeNovoAlgorithm "OpenMS::DeNovoAlgorithm":
    # wrap-ignore
    DeNovoAlgorithm()
    DeNovoAlgorithm(DeNovoAlgorithm)
    void generateCandidates(libcpp_vector[ PeptideIdentification ] & candidates,\
       libcpp_vector[ libcpp_vector[ IonScore_DeNovoIonScoring ] ] &ion_scores,\
       MSExperiment[RichPeak1D, ChromatogramPeak] &exp)
    void generateCandidates(PeptideIdentification &candidates, libcpp_vector[\
       IonScore_DeNovoIonScoring ] & ion_scores, MSSpectrum[RichPeak1D] & spec)
```

6.78 DeNovoIdentification

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/DENOVO/DeNovoIdentification.h>" namespace\
       "OpenMS":

  cdef cppclass DeNovoIdentification "OpenMS::DeNovoIdentification":
    # wrap-ignore
    DeNovoIdentification()
    DeNovoIdentification(DeNovoIdentification)
    void getIdentifications(libcpp_vector[ PeptideIdentification ] & ids,\
       MSExperiment[RichPeak1D, ChromatogramPeak] & exp)
    void getIdentification(PeptideIdentification & id_, MSSpectrum[RichPeak1D] &\
```

```
       spectrum)
```

6.79 DeNovoIonScoring

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/DENOVO/DeNovoIonScoring.h>" namespace "OpenMS":

  cdef cppclass DeNovoIonScoring "OpenMS::DeNovoIonScoring":
    # wrap-ignore
    DeNovoIonScoring()
    DeNovoIonScoring(DeNovoIonScoring)
    void getIonScores(libcpp_vector[ IonScore_DeNovoIonScoring ] &ion_scores,\
       MSSpectrum[RichPeak1D] &spec)
    void getIonScores(libcpp_vector[ libcpp_vector[ IonScore_DeNovoIonScoring ] ]\
       &ion_scores, MSExperiment[RichPeak1D, ChromatogramPeak] &exp)  #wrap-ignore

cdef extern from "<OpenMS/ANALYSIS/DENOVO/DeNovoIonScoring.h>" namespace\
       "OpenMS::DeNovoIonScoring":

  cdef cppclass IonScore_DeNovoIonScoring "OpenMS::DeNovoIonScoring::IonScore":
    IonScore_DeNovoIonScoring()
    IonScore_DeNovoIonScoring(IonScore_DeNovoIonScoring)

    double score

    double position

    ptrdiff_t index
```

6.80 DeNovoPostScoring

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/DENOVO/DeNovoPostScoring.h>" namespace "OpenMS":

  cdef cppclass DeNovoPostScoring "OpenMS::DeNovoPostScoring":
    # wrap-ignore
    DeNovoPostScoring()
    DeNovoPostScoring(DeNovoPostScoring)
    void apply(libcpp_vector[ PeptideIdentification ] &identifications,\
       MSExperiment[RichPeak1D, ChromatogramPeak] &exp)
```

```
    void apply(PeptideIdentification &identification, MSSpectrum[RichPeak1D] &spec)
```

6.81 DefaultParamHandler

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/DATASTRUCTURES/DefaultParamHandler.h>" namespace\
       "OpenMS":

  cdef cppclass DefaultParamHandler:
    #wrap-ignore

    void setParameters(Param &param)
    Param getParameters()
    Param getDefaults()
    String getName()
    void setName(String)
```

6.82 DigestSimulation

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/SIMULATION/DigestSimulation.h>" namespace "OpenMS":

  cdef cppclass DigestSimulation "OpenMS::DigestSimulation":
    DigestSimulation()
    DigestSimulation(DigestSimulation)
    void digest(FeatureMap & feature_map)
```

6.83 Digestion

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/Digestion.h>" namespace "OpenMS":

  cdef cppclass Digestion "OpenMS::Digestion":
    Digestion()
    Digestion(Digestion)
    String  getEnzyme()
    void setEnzyme(String &enzyme)
```

```
    double getDigestionTime()
    void setDigestionTime(double digestion_time)
    double getTemperature()
    void setTemperature(double temperature)
    double getPh()
    void setPh(double ph)
```

6.84 DistanceMatrix

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/DATASTRUCTURES/DistanceMatrix.h>" namespace "OpenMS":

  cdef cppclass DistanceMatrix[Value]:
    # wrap-instances:
    #   DistanceMatrix := DistanceMatrix[float]
    DistanceMatrix()
    DistanceMatrix(DistanceMatrix)
    DistanceMatrix(size_t dimensionsize, Value value)
    # ValueType operator()(size_t i, size_t j)
    # ValueType operator()(size_t i, size_t j)
    Value getValue(size_t i, size_t j)
    void setValue(size_t i, size_t j, Value value)
    void setValueQuick(size_t i, size_t j, Value value)
    void clear()
    void resize(size_t dimensionsize, Value value)
    void reduce(size_t j)
    size_t dimensionsize()
    void updateMinElement()
    bool operator==(DistanceMatrix[ Value ] &rhs)
    libcpp_pair[ size_t, size_t ] getMinElementCoordinates()
    # TEMPLATE # std::ostream  operator[[(std::ostream &os, DistanceMatrix[ Value ]\
       &matrix)
```

6.85 DocumentIdentifier

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/DocumentIdentifier.h>" namespace "OpenMS":

  cdef cppclass DocumentIdentifier:
```

```
    DocumentIdentifier()
    DocumentIdentifier(DocumentIdentifier)  # wrap-ignore

    void setIdentifier(String id)

    String getIdentifier()

    void setLoadedFileType(String file_name)

    int getLoadedFileType()

    void setLoadedFilePath(String file_name)
    String getLoadedFilePath()
```

6.86 DoubleList

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
ctypedef libcpp_vector[ double ] DoubleList
```

6.87 EDTAFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/EDTAFile.h>" namespace "OpenMS":

  cdef cppclass EDTAFile:

    EDTAFile()
    EDTAFile(EDTAFile)

    void store(String filename, FeatureMap & map)
    void store(String filename, ConsensusMap & map)
    void load(String filename, ConsensusMap & consensus_map)
```

6.88 Element

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/Element.h>" namespace "OpenMS":

  cdef cppclass Element:

    Element()
    Element(Element)  # wrap-ignore

    Element(String name,
        String symbol,
        UInt atomic_number,
        double average_weight,
        double mono_weight,
        IsotopeDistribution isotopes)

    void setAtomicNumber(UInt atomic_number)

    UInt getAtomicNumber()

    void setAverageWeight(double weight)

    double getAverageWeight()

    void setMonoWeight(double weight)

    double getMonoWeight()

    void setIsotopeDistribution(IsotopeDistribution isotopes)

    IsotopeDistribution getIsotopeDistribution()

    void setName(String name)

    String getName()

    void setSymbol(String symbol)

    String getSymbol()
```

6.89 ElementDB

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/ElementDB.h>" namespace "OpenMS":

  cdef cppclass ElementDB "OpenMS::ElementDB":
```

```
    # wrap-manual-memory

    ElementDB(ElementDB)  #wrap-ignore

    const Element * getElement(String & name)
    const Element * getElement(UInt atomic_number)
    bool hasElement(String & name)
    bool hasElement(UInt atomic_number)

cdef extern from "<OpenMS/CHEMISTRY/ElementDB.h>" namespace "OpenMS::ElementDB":

  const ElementDB* getInstance()  # wrap-ignore
```

6.90 ElutionPeakDetection

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/DATAREDUCTION/ElutionPeakDetection.h>" namespace\
       "OpenMS":

  cdef cppclass ElutionPeakDetection(ProgressLogger, DefaultParamHandler):
    # wrap-inherits:
    #  ProgressLogger
    #  DefaultParamHandler

    ElutionPeakDetection()

    void detectPeaks(Kernel_MassTrace & in_,
             libcpp_vector[Kernel_MassTrace] & out
             )

    void detectPeaks(libcpp_vector[Kernel_MassTrace] & in_,
             libcpp_vector[Kernel_MassTrace] & out
            )

    void filterByPeakWidth(libcpp_vector[Kernel_MassTrace] & in_,
                 libcpp_vector[Kernel_MassTrace] & out
                )

    double computeMassTraceNoise(Kernel_MassTrace &)
    double computeMassTraceSNR(Kernel_MassTrace &)
    double computeApexSNR(Kernel_MassTrace &)

    void findLocalExtrema(Kernel_MassTrace & , Size & , libcpp_vector[ size_t ] & ,\
       libcpp_vector[ size_t ] & )
```

```
    void smoothData(Kernel_MassTrace & mt, int win_size)
```

6.91 EmgFitter1D

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/EmgFitter1D.h>" namespace\
       "OpenMS":

  cdef cppclass EmgFitter1D(LevMarqFitter1D):
    # wrap-inherits:
    #  LevMarqFitter1D
    EmgFitter1D()
    EmgFitter1D(EmgFitter1D)
    # float fit1d(libcpp_vector[Peak1D] range_, InterpolationModel * & model)  #\
       wrap-ignore
    # Fitter1D * create()
    String getProductName()
```

6.92 EmgModel

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/EmgModel.h>" namespace\
       "OpenMS":

  cdef cppclass EmgModel(InterpolationModel):
    # wrap-inherits:
    #  InterpolationModel
    EmgModel()
    EmgModel(EmgModel)
    # BaseModel[ 1 ] * create()
    String getProductName()

    # void setOffset(CoordinateType offset) # wrap-ignore
    # void setSamples() # wrap-ignore
    # CoordinateType getCenter() # wrap-ignore
```

6.93 EmgScoring

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/EmgScoring.h>" namespace\
       "OpenMS":

  cdef cppclass EmgScoring "OpenMS::EmgScoring":
    EmgScoring()
    EmgScoring(EmgScoring)  #wrap-ignore
    void setFitterParam(Param param)
    Param getDefaults()
    # TEMPLATE # double calcElutionFitScore(MRMFeature &mrmfeature,\
       MRMTransitionGroup[ SpectrumType, TransitionT ] &transition_group)
    double elutionModelFit(libcpp_vector[DPosition2] current_section, bool\
       smooth_data)
```

6.94 EmpiricalFormula

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/EmpiricalFormula.h>" namespace "OpenMS":

  cdef cppclass EmpiricalFormula:

    EmpiricalFormula()
    EmpiricalFormula(EmpiricalFormula)  # wrap-ignore

    EmpiricalFormula(String)

    EmpiricalFormula(SignedSize number, Element * element, SignedSize charge)

    double getMonoWeight()

    double getAverageWeight()

    IsotopeDistribution getIsotopeDistribution(UInt max_depth)

    Size getNumberOf(Element * element)

    Size getNumberOfAtoms()

    SignedSize getCharge()
```

```
    void setCharge(SignedSize charge)

    String toString()

    bool isEmpty()

    bool isCharged()

    bool hasElement(Element * element)

    bool contains(EmpiricalFormula ef)

    bool operator==(EmpiricalFormula)

    bool operator!=(EmpiricalFormula)
```

6.95 EnzymaticDigestion

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/EnzymaticDigestion.h>" namespace "OpenMS":

  cdef cppclass EnzymaticDigestion:

    EnzymaticDigestion()

    SignedSize getMissedCleavages()
    void setMissedCleavages(SignedSize missed_cleavages)

    String getEnzymeName()
    void setEnzyme(String name)

    void digest(AASequence & protein, libcpp_vector[AASequence] & output)
    Size peptideCount(AASequence & protein)

    Specificity getSpecificity()

    void setSpecificity(Specificity spec)

    Specificity getSpecificityByName(String name)

    bool isValidProduct(AASequence protein, Size pep_pos, Size pep_length)

cdef extern from "<OpenMS/CHEMISTRY/EnzymaticDigestion.h>" namespace\
       "OpenMS::EnzymaticDigestion":
```

```
  #  # wrap-attach:
  #  #  EnzymaticDigestion
  #  TRYPSIN,
  #  SIZE_OF_TRYPSIN

  cdef enum Specificity:
    # wrap-attach:
    #  EnzymaticDigestion
    SPEC_FULL,  # fully enzyme specific, e.g., tryptic (ends with KR, AA-before is\
       KR), or peptide is at protein terminal ends
    SPEC_SEMI,  # semi specific, i.e., one of the two cleavage sites must fulfill\
       requirements
    SPEC_NONE,  # no requirements on start / end
    SIZE_OF_SPECIFICITY
```

6.96 EnzymaticDigestionLogModel

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/EnzymaticDigestionLogModel.h>" namespace\
       "OpenMS":

  cdef cppclass EnzymaticDigestionLogModel:

    EnzymaticDigestionLogModel()

    String getEnzymeName()
    void setEnzyme(String name)

    double getLogThreshold()
    void setLogThreshold(double threshold)

    void digest(AASequence & protein, libcpp_vector[AASequence] & output)
    Size peptideCount(AASequence & protein)
```

6.97 Enzyme

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/Enzyme.h>" namespace "OpenMS":
```

```
cdef cppclass Enzyme:
  Enzyme(Enzyme)  # wrap-ignore

  Enzyme(String name,
       String cleavage_regex,
       libcpp_set[String]  synonyms,
       String regex_description,
       EmpiricalFormula n_term_gain,
       EmpiricalFormula c_term_gain,
       String psi_id,
       String xtandem_id,
       UInt omssa_id)

  void setName(String name)

  String getName()

  void setSynonyms(libcpp_set[String] synonyms)

  void addSynonym(String synonym)

  libcpp_set[String] getSynonyms()

  void setRegEx(String three_letter_code)

  String getRegEx()

  void setRegExDescription(String one_letter_code)

  String getRegExDescription()

  void setNTermGain(EmpiricalFormula value)

  void setCTermGain(EmpiricalFormula value)

  EmpiricalFormula getNTermGain()

  EmpiricalFormula getCTermGain()

  void setPSIid(String value)

  String getPSIid()

  void setXTANDEMid(String value)
  String getXTANDEMid()

  void setOMSSAid(int value)
```

```
    int getOMSSAid()

    bool operator==(Enzyme & Enzyme)

    bool operator!=(Enzyme & Enzyme)

    bool operator==(EmpiricalFormula cleavage_regex)

    bool operator!=(EmpiricalFormula cleavage_regex)
```

6.98 EnzymesDB

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/EnzymesDB.h>" namespace "OpenMS":

  cdef cppclass EnzymesDB "OpenMS::EnzymesDB":
    # wrap-manual-memory

    EnzymesDB()  #wrap-ignore
    EnzymesDB(EnzymesDB)  #wrap-ignore

    const Enzyme * getEnzyme(String & name)
    const Enzyme * getEnzymeByRegEx(String & cleavage_regex)
    void setEnzymes(String & filename)
    void addEnzyme(Enzyme & enzyme)
    void clear()
    void getAllNames(libcpp_vector[ String ] & all_names)
    void getAllXTandemNames(libcpp_vector[ String ] & all_names)
    void getAllOMSSANames(libcpp_vector[ String ] & all_names)
    bool hasEnzyme(String & name)
    bool hasRegEx(String & cleavage_regex)
cdef extern from "<OpenMS/CHEMISTRY/EnzymesDB.h>" namespace "OpenMS::EnzymesDB":

  EnzymesDB* getInstance()  # wrap-ignore
```

6.99 ExperimentalSettings

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/ExperimentalSettings.h>" namespace "OpenMS":

  cdef cppclass ExperimentalSettings(MetaInfoInterface,DocumentIdentifier):
```

```
    # wrap-inherits:
    #  DocumentIdentifier
    #  MetaInfoInterface

    ExperimentalSettings()
    ExperimentalSettings(ExperimentalSettings)  # wrap-ignore

    libcpp_vector[SourceFile] getSourceFiles()
    void setSourceFiles(libcpp_vector[SourceFile] source_files)

    DateTime getDateTime()
    void setDateTime(DateTime date_time)

    Sample getSample()
    void setSample(Sample sample)

    libcpp_vector[ContactPerson] getContacts()
    void setContacts(libcpp_vector[ContactPerson] contacts)

    Instrument getInstrument()
    void setInstrument(Instrument instrument)

    HPLC getHPLC()
    void setHPLC(HPLC hplc)

    String getComment()
    void setComment(String comment)

    libcpp_vector[ProteinIdentification] getProteinIdentifications()
    void setProteinIdentifications(libcpp_vector[ProteinIdentification]\
       protein_identifications)

    String getFractionIdentifier()
    void setFractionIdentifier(String fraction_identifier)
```

6.100 FASTAFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/FASTAFile.h>" namespace "OpenMS":

  cdef cppclass FASTAFile:

    FASTAFile()
    FASTAFile(FASTAFile)  #wrap-ignore
```

```
    void load(String& filename, libcpp_vector[FASTAEntry] & data)
    void store(String& filename, libcpp_vector[FASTAEntry] & data)

cdef extern from "<OpenMS/FORMAT/FASTAFile.h>" namespace "OpenMS::FASTAFile":

  cdef cppclass FASTAEntry:
    FASTAEntry()
    FASTAEntry(FASTAEntry)

    String identifier
    String description
    String sequence
```

6.101 FalseDiscoveryRate

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/ID/FalseDiscoveryRate.h>" namespace "OpenMS":

  cdef cppclass FalseDiscoveryRate(DefaultParamHandler):
    # wrap-inherits:
    #  DefaultParamHandler

    FalseDiscoveryRate()
    FalseDiscoveryRate(FalseDiscoveryRate)    #wrap-ignore

    void apply(libcpp_vector[PeptideIdentification] & forward_ids,\
       libcpp_vector[PeptideIdentification] & reverse_ids)
    void apply(libcpp_vector[PeptideIdentification] & id)
    void apply(libcpp_vector[ProteinIdentification] & forward_ids,\
       libcpp_vector[ProteinIdentification] & reverse_ids)
    void apply(libcpp_vector[ProteinIdentification] & id)
```

6.102 Feature

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/KERNEL/Feature.h>" namespace "OpenMS":

  cdef cppclass Feature(UniqueIdInterface):
    # wrap-inherits:
```

```
#  UniqueIdInterface

Feature()
Feature(Feature &)

void setMZ(double)
void setRT(double)
void setIntensity(double)

double getMZ()
double getRT()
double getIntensity()

float getQuality(Size index)
void setQuality(Size index, float q)
float getOverallQuality()
void setOverallQuality(float q)

float getWidth()
void setWidth(float q)

Int getCharge()
void setCharge(Int q)

libcpp_vector[Feature] getSubordinates()
void setSubordinates(libcpp_vector[Feature])

bool encloses(double rt, double mz)
ConvexHull2D getConvexHull()
libcpp_vector[ConvexHull2D] getConvexHulls()
void setConvexHulls(libcpp_vector[ConvexHull2D])

libcpp_vector[PeptideIdentification] getPeptideIdentifications()
void setPeptideIdentifications(libcpp_vector[PeptideIdentification] & peptides)

bool operator==(Feature)
bool operator!=(Feature)

void getKeys(libcpp_vector[String] & keys)
void getKeys(libcpp_vector[unsigned int] & keys)  # wrap-as:getKeysAsIntegers
DataValue getMetaValue(unsigned int)
DataValue getMetaValue(String)
void setMetaValue(unsigned int, DataValue)
void setMetaValue(String, DataValue)
bool metaValueExists(String)
bool metaValueExists(unsigned int)
void removeMetaValue(String)
void removeMetaValue(unsigned int)
```

```
    void clearMetaInfo()
```

6.103 FeatureDeconvolution

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/DECHARGING/FeatureDeconvolution.h>" namespace\
       "OpenMS":

  cdef cppclass FeatureDeconvolution(DefaultParamHandler):
    # wrap-inherits:
    #  DefaultParamHandler

    FeatureDeconvolution()
    FeatureDeconvolution(FeatureDeconvolution)    #wrap-ignore

    void compute(FeatureMap & input, FeatureMap & output, ConsensusMap & cmap1,\
       ConsensusMap & cmap2)

cdef extern from "<OpenMS/ANALYSIS/DECHARGING/FeatureDeconvolution.h>" namespace\
       "OpenMS::FeatureDeconvolution":

  cdef enum CHARGEMODE "OpenMS::FeatureDeconvolution::CHARGEMODE":
    QFROMFEATURE
    QHEURISTIC
    QALL
```

6.104 FeatureDistance

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/MAPMATCHING/FeatureDistance.h>" namespace\
       "OpenMS":

  cdef cppclass FeatureDistance(DefaultParamHandler) :
    # wrap-inherits:
    #  DefaultParamHandler
    FeatureDistance(FeatureDistance)  #wrap-ignore
    # TODO  is static const -> no setters please
    # double infinity
    FeatureDistance(double max_intensity, bool force_constraints)
    # libcpp_pair[ bool, double ] operator()(BaseFeature & left, BaseFeature &\
       right)
```

6.105 FeatureFileOptions

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/OPTIONS/FeatureFileOptions.h>" namespace "OpenMS":

  cdef cppclass FeatureFileOptions:

    FeatureFileOptions()
    FeatureFileOptions(FeatureFileOptions)

    void setMetadataOnly(bool)
    bool getMetadataOnly()

    void setSizeOnly(bool)
    bool getSizeOnly()

    void setLoadConvexHull(bool)
    bool getLoadConvexHull()

    void setLoadSubordinates(bool)
    bool getLoadSubordinates()

    void setRTRange(DRange1 & range_)
    bool hasRTRange()
    DRange1 getRTRange()
    void setMZRange(DRange1 & range_)
    bool hasMZRange()
    DRange1 getMZRange()
    void setIntensityRange(DRange1 & range_)
    bool hasIntensityRange()
    DRange1 getIntensityRange()
```

6.106 FeatureFinder

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/FeatureFinder.h>" namespace\
       "OpenMS":

  cdef cppclass FeatureFinder(ProgressLogger):
```

```
    # wrap-inherits:
    #  ProgressLogger
    #

    FeatureFinder()
    void run(String algorithm_name,
         MSExperiment[Peak1D, ChromatogramPeak] & input_map,
         FeatureMap & feats,
         Param & param,
         FeatureMap & seeds
         )

    Param getParameters(String algorithm_name)
```

6.107 FeatureFinderAlgorithmIsotopeWavelet

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from\
       "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/FeatureFinderAlgorithmIsotopeWavelet.h>"\
       namespace "OpenMS":

  cdef cppclass FeatureFinderAlgorithmIsotopeWavelet(DefaultParamHandler):

    # wrap-inherits:
    #  DefaultParamHandler
    FeatureFinderAlgorithmIsotopeWavelet()

    void setData(MSExperiment[Peak1D, ChromatogramPeak] & input, FeatureMap& output,\
       FeatureFinder & ff)
    void run()
cdef extern from\
       "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/FeatureFinderAlgorithmIsotopeWavelet.h>"\
       namespace "OpenMS::FeatureFinderAlgorithmIsotopeWavelet":

  String getProductName()    # wrap-attach:FeatureFinderAlgorithmIsotopeWavelet
```

6.108 FeatureFinderAlgorithmPicked

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from\
       "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/FeatureFinderAlgorithmPicked.h>" namespace\
       "OpenMS":
```

```
  cdef cppclass FeatureFinderAlgorithmPicked(DefaultParamHandler):

    # wrap-inherits:
    #  DefaultParamHandler
    FeatureFinderAlgorithmPicked()

    void setData(MSExperiment[Peak1D, ChromatogramPeak] & input, FeatureMap &\
       output, FeatureFinder & ff)
    void run()

    void setSeeds(FeatureMap& seeds)

cdef extern from\
       "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/FeatureFinderAlgorithmPicked.h>" namespace\
       "OpenMS::FeatureFinderAlgorithmPicked":

  String getProductName()    # wrap-attach:FeatureFinderAlgorithmPicked
```

6.109 FeatureFinderAlgorithmPickedHelperStructs

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from\
       "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/FeatureFinderAlgorithmPickedHelperStructs.h>"\
       namespace "OpenMS::FeatureFinderAlgorithmPickedHelperStructs":

  cdef cppclass TheoreticalIsotopePattern\
       "OpenMS::FeatureFinderAlgorithmPickedHelperStructs::TheoreticalIsotopePattern":
    TheoreticalIsotopePattern()  # wrap-ignore
    TheoreticalIsotopePattern(TheoreticalIsotopePattern)  #wrap-ignore

    libcpp_vector[ double ] intensity
    Size optional_begin
    Size optional_end
    double max
    Size trimmed_left

    Size size()

cdef extern from\
       "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/FeatureFinderAlgorithmPickedHelperStructs.h>"\
       namespace "OpenMS::FeatureFinderAlgorithmPickedHelperStructs":

  cdef cppclass MassTrace:
    MassTrace(MassTrace)  #wrap-ignore
```

```
    # POINTER # PeakType * max_peak
    double max_rt
    double theoretical_int
    # POINTER # libcpp_vector[ libcpp_pair[ double, PeakType * ] ] peaks
    ConvexHull2D getConvexhull()
    void updateMaximum()
    double getAvgMZ()
    bool isValid()

  cdef cppclass MassTraces:
    MassTraces()
    MassTraces(MassTraces)  #wrap-ignore
    Size max_trace
    double baseline
    Size getPeakCount()
    bool isValid(double seed_mz, double trace_tolerance)
    Size getTheoreticalmaxPosition()
    void updateBaseline()
    libcpp_pair[ double, double ] getRTBounds()

cdef extern from\
       "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/FeatureFinderAlgorithmPickedHelperStructs.h>"\
       namespace "OpenMS::FeatureFinderAlgorithmPickedHelperStructs":

  cdef cppclass Seed "OpenMS::FeatureFinderAlgorithmPickedHelperStructs::Seed":
    Seed(Seed)  #wrap-ignore
    Size spectrum
    Size peak
    float intensity
    bool operator<(Seed & rhs)

cdef extern from\
       "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/FeatureFinderAlgorithmPickedHelperStructs.h>"\
       namespace "OpenMS::FeatureFinderAlgorithmPickedHelperStructs":

  cdef cppclass IsotopePattern\
       "OpenMS::FeatureFinderAlgorithmPickedHelperStructs::IsotopePattern":
    IsotopePattern(IsotopePattern)  #wrap-ignore
    # TODO STL attributes -- Signed size does not work either!
    # vector.from_py:33:13: 'ptrdiff_t' is not a type identifier
    # libcpp_vector[ SignedSize ] peak
    libcpp_vector[ size_t ] spectrum
    libcpp_vector[ double ] intensity
    libcpp_vector[ double ] mz_score
    libcpp_vector[ double ] theoretical_mz
    TheoreticalIsotopePattern theoretical_pattern
    IsotopePattern(Size size)
```

6.110 FeatureFinderAlgorithmSH

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/FeatureFinderAlgorithmSH.h>"\
       namespace "OpenMS":

  cdef cppclass FeatureFinderAlgorithmSH(DefaultParamHandler):

    # wrap-inherits:
    #  DefaultParamHandler
    FeatureFinderAlgorithmSH()

    void setData(MSExperiment[Peak1D, ChromatogramPeak] & input, FeatureMap &\
       output, FeatureFinder & ff)
    void run()
    unsigned int getNativeScanId(String native_id)

cdef extern from "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/FeatureFinderAlgorithmSH.h>"\
       namespace "OpenMS::FeatureFinderAlgorithmSH":

  String getProductName()    # wrap-attach:FeatureFinderAlgorithmSH
```

6.111 FeatureFindingMetabo

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/DATAREDUCTION/FeatureFindingMetabo.h>" namespace\
       "OpenMS":

  cdef cppclass FeatureFindingMetabo(ProgressLogger, DefaultParamHandler):
    # wrap-inherits:
    #  ProgressLogger
    #  DefaultParamHandler
    #

    FeatureFindingMetabo()

    void run(libcpp_vector[Kernel_MassTrace] input,
         FeatureMap & result
         )
```

6.112 FeatureGroupingAlgorithm

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/MAPMATCHING/FeatureGroupingAlgorithm.h>"\
       namespace "OpenMS":

  cdef cppclass FeatureGroupingAlgorithm(DefaultParamHandler):
    # wrap-inherits:
    #  DefaultParamHandler
    # wrap-ignore
    # Abstract Class
    void transferSubelements(libcpp_vector[ConsensusMap] maps,
                 ConsensusMap & out
                )

    void registerChildren()
```

6.113 FeatureGroupingAlgorithmIdentification

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from\
       "<OpenMS/ANALYSIS/MAPMATCHING/FeatureGroupingAlgorithmIdentification.h>" namespace\
       "OpenMS":

  cdef cppclass FeatureGroupingAlgorithmIdentification(FeatureGroupingAlgorithm) :
    # wrap-inherits:
    #  FeatureGroupingAlgorithm
    FeatureGroupingAlgorithmIdentification()
    FeatureGroupingAlgorithmIdentification(FeatureGroupingAlgorithmIdentification) \
       #wrap-ignore
    void group(libcpp_vector[ FeatureMap ] & maps, ConsensusMap & out)
    # POINTER # FeatureGroupingAlgorithm * create()
    String getProductName()
```

6.114 FeatureGroupingAlgorithmLabeled

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/MAPMATCHING/FeatureGroupingAlgorithmLabeled.h>"\
       namespace "OpenMS":
```

```
  cdef cppclass FeatureGroupingAlgorithmLabeled(FeatureGroupingAlgorithm) :
    # wrap-inherits:
    #  FeatureGroupingAlgorithm
    FeatureGroupingAlgorithmLabeled()
    FeatureGroupingAlgorithmLabeled(FeatureGroupingAlgorithmLabeled)  #wrap-ignore
    void group(libcpp_vector[ FeatureMap ] & maps, ConsensusMap & out)
    # POINTER # FeatureGroupingAlgorithm * create()
    String getProductName()
```

6.115 FeatureGroupingAlgorithmQT

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/MAPMATCHING/FeatureGroupingAlgorithmQT.h>"\
       namespace "OpenMS":

  cdef cppclass FeatureGroupingAlgorithmQT(FeatureGroupingAlgorithm):
    # wrap-inherits:
    #  FeatureGroupingAlgorithm

    FeatureGroupingAlgorithmQT()

    void group(libcpp_vector[FeatureMap] & maps,
           ConsensusMap & out
          )

    void group(libcpp_vector[ConsensusMap] & maps,
           ConsensusMap & out
          )

    String getProductName()
```

6.116 FeatureGroupingAlgorithmUnlabeled

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/MAPMATCHING/FeatureGroupingAlgorithmUnlabeled.h>"\
       namespace "OpenMS":

  cdef cppclass FeatureGroupingAlgorithmUnlabeled(FeatureGroupingAlgorithm) :
    # wrap-inherits:
    #  FeatureGroupingAlgorithm
```

```
    FeatureGroupingAlgorithmUnlabeled()
    FeatureGroupingAlgorithmUnlabeled(FeatureGroupingAlgorithmUnlabeled) \
       #wrap-ignore
    void group(libcpp_vector[ FeatureMap ] & maps, ConsensusMap & out)
    # POINTER # FeatureGroupingAlgorithm * create()
    String getProductName()
    void addToGroup(int map_id, FeatureMap feature_map)
    void setReference(int map_id, FeatureMap map)
    ConsensusMap getResultMap()
```

6.117 FeatureHandle

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/KERNEL/FeatureHandle.h>" namespace "OpenMS":

  cdef cppclass FeatureHandle(Peak2D,UniqueIdInterface) :
    # wrap-inherits:
    #  Peak2D
    #  UniqueIdInterface

    FeatureHandle()
    FeatureHandle(FeatureHandle)

    FeatureHandle(UInt64 map_index, Peak2D & point, UInt64 element_index)
    UInt64 getMapIndex()
    void setMapIndex(UInt64 i)
    void setCharge(Int charge)
    Int getCharge()
    void setWidth(float width)
    float getWidth()

    bool operator==(FeatureHandle & i)
    bool operator!=(FeatureHandle & i)
```

6.118 FeatureMap

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/KERNEL/FeatureMap.h>" namespace "OpenMS":

  cdef cppclass FeatureMap(UniqueIdInterface, DocumentIdentifier, RangeManager2):
```

```
# wrap-inherits:
#   UniqueIdInterface
#   DocumentIdentifier
#   RangeManager2
#
# wrap-instances:
#   FeatureMap := FeatureMap

FeatureMap()
FeatureMap(FeatureMap &)

bool operator==(FeatureMap)
bool operator!=(FeatureMap)

int size()
Feature operator[](int)     #wrap-upper-limit:size()
void push_back(Feature spec)

void sortByIntensity()
void sortByIntensity(bool reverse)
void sortByPosition()
void sortByRT()
void sortByMZ()
void sortByOverallQuality()

void swap(FeatureMap &)
void swapFeaturesOnly(FeatureMap swapfrom)
void clear()
void clear(bool clear_meta_data)

FeatureMap operator+(FeatureMap)
FeatureMap iadd(FeatureMap)  # wrap-as:operator+=

void   updateRanges()

libcpp_vector[ProteinIdentification] getProteinIdentifications() nogil except+
void setProteinIdentifications(libcpp_vector[ProteinIdentification]) nogil\
   except+

libcpp_vector[PeptideIdentification] getUnassignedPeptideIdentifications() nogil\
   except+
void setUnassignedPeptideIdentifications(libcpp_vector[PeptideIdentification])\
   nogil except+

Size applyMemberFunction(Size(* fun)()) # wrap-ignore

libcpp_vector[DataProcessing] getDataProcessing()
```

```
    void setDataProcessing(libcpp_vector[DataProcessing])

    void setPrimaryMSRunPath(StringList& s)
    StringList getPrimaryMSRunPath()

    libcpp_vector[Feature].iterator begin()   # wrap-iter-begin:__iter__(Feature)
    libcpp_vector[Feature].iterator end()     # wrap-iter-end:__iter__(Feature)
```

6.119 FeatureXMLFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/FeatureXMLFile.h>" namespace "OpenMS":

  cdef cppclass FeatureXMLFile:
    FeatureXMLFile()

    void load(String, FeatureMap &) nogil except+
    void store(String, FeatureMap &) nogil except+

    FeatureFileOptions getOptions()
    void setOptions(FeatureFileOptions)

    Size loadSize(String path)
```

6.120 File

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/SYSTEM/File.h>" namespace "OpenMS":

  cdef cppclass File:
    pass

cdef extern from "<OpenMS/SYSTEM/File.h>" namespace "OpenMS::File":

  String getExecutablePath()  # wrap-attach:File

  bool exists(String file_)  # wrap-attach:File

  bool empty(String file_)  # wrap-attach:File
```

```
bool remove(String file_)  # wrap-attach:File

bool removeDirRecursively(String dir_name)  # wrap-attach:File

String absolutePath(String file)  # wrap-attach:File

String basename(String file)  # wrap-attach:File

String path(String file)  # wrap-attach:File

String removeExtension(String file)  # wrap-attach:File

bool readable(String file)  # wrap-attach:File

bool writable(String file)  # wrap-attach:File

bool isDirectory(String path)  # wrap-attach:File

String find(String filename, StringList directories)  # wrap-attach:File

bool fileList(String dir, String file_pattern, StringList output, bool full_path) \
     # wrap-attach:File

String getUniqueName()  # wrap-attach:File

String getOpenMSDataPath()  # wrap-attach:File

String getOpenMSHomePath()  # wrap-attach:File

String getTempDirectory()  # wrap-attach:File

String getUserDirectory()  # wrap-attach:File

Param getSystemParameters()  # wrap-attach:File

String findDatabase(String db_name)  # wrap-attach:File

String findExecutable(String toolName)  # wrap-attach:File

String getTemporaryFile(String & alternative_file)  # wrap-attach:File

String findDoc(String filename)  # wrap-attach:File
```

6.121 FileHandler

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/FileHandler.h>" namespace "OpenMS":

  cdef cppclass FileHandler:  # wrap=True
    FileHandler()
    FileHandler(FileHandler)

    bool loadExperiment(libcpp_string, MSExperiment[Peak1D, ChromatogramPeak] &)\
       nogil except+
    void storeExperiment(libcpp_string, MSExperiment[Peak1D, ChromatogramPeak])\
       nogil except+
    bool loadFeatures(libcpp_string, FeatureMap &)

    PeakFileOptions  getOptions()
    void setOptions(PeakFileOptions)

cdef extern from "<OpenMS/FORMAT/FileHandler.h>" namespace "OpenMS::FileHandler":

  int  getType(String filename)  # wrap-attach:FileHandler
  Type getTypeByFileName(String & filename)  # wrap-attach:FileHandler
  Type getTypeByContent(String & filename)  # wrap-attach:FileHandler
  String computeFileHash(String & filename)  # wrap-attach:FileHandler
  bool isSupported(Type type_)  # wrap-attach:FileHandler
```

6.122 FileTypes

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/FileTypes.h>" namespace "OpenMS::FileTypes":

  cdef enum Type:

      UNKNOWN,
      DTA,
      DTA2D,
      MZDATA,
      MZXML,
      FEATUREXML,
      IDXML,
      CONSENSUSXML,
      MGF,
      INI,
      TOPPAS,
      TRANSFORMATIONXML,
      MZML,
      MS2,
```

```
      PEPXML,
      PROTXML,
      MZIDENTML,
      GELML,
      TRAML,
      MSP,
      OMSSAXML,
      MASCOTXML,
      PNG,
      XMASS,
      TSV,
      PEPLIST,
      HARDKLOER,
      KROENIK,
      FASTA,
      EDTA
      SIZE_OF_TYPE
```

6.123 FilterFunctor

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/TRANSFORMERS/FilterFunctor.h>" namespace\
       "OpenMS":

  cdef cppclass FilterFunctor(DefaultParamHandler) :
    # wrap-inherits:
    #  DefaultParamHandler
    FilterFunctor()
    FilterFunctor(FilterFunctor)
    # double apply(MSSpectrum[Peak1D] & )
    void registerChildren()
```

6.124 Fitter1D

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
ctypedef libcpp_vector[Peak1D] RawDataArrayType

cdef extern from "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/Fitter1D.h>" namespace\
       "OpenMS":

  cdef cppclass Fitter1D(DefaultParamHandler):
```

```
    # wrap-inherits:
    #  DefaultParamHandler
    Fitter1D()
    Fitter1D(Fitter1D)
    # QualityType fit1d(RawDataArrayType &, InterpolationModel *&)
    void registerChildren()
```

6.125 GaussFilter

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/SMOOTHING/GaussFilter.h>" namespace "OpenMS":

  cdef cppclass GaussFilter(DefaultParamHandler,ProgressLogger):
    # wrap-inherits:
    #  DefaultParamHandler
    #  ProgressLogger

    GaussFilter()
    GaussFilter(GaussFilter)

    void filter(MSSpectrum[Peak1D] & spectrum)
    void filter(MSChromatogram[ChromatogramPeak] & chromatogram)
    void filterExperiment(MSExperiment[Peak1D,ChromatogramPeak] & exp)
```

6.126 GaussFitter

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/MATH/STATISTICS/GaussFitter.h>" namespace "OpenMS::Math":

  cdef cppclass GaussFitter:

    GaussFitter()
    GaussFitter(GaussFitter)    # wrap-ignore

    void setInitialParameters(GaussFitResult & result)

    GaussFitResult fit(libcpp_vector[DPosition2] points)

cdef extern from "<OpenMS/MATH/STATISTICS/GaussFitter.h>" namespace\
       "OpenMS::Math::GaussFitter":
```

```
  cdef cppclass GaussFitResult:

    GaussFitResult()
    GaussFitResult(double, double, double)
    GaussFitResult(GaussFitResult)    # wrap-ignore

    double A
    double x0
    double sigma
```

6.127 GaussTraceFitter

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/GaussTraceFitter.h>"\
       namespace "OpenMS":

  cdef cppclass GaussTraceFitter:
    GaussTraceFitter()
    GaussTraceFitter(GaussTraceFitter)
    void fit(MassTraces& traces)
    double getLowerRTBound()
    double getUpperRTBound()
    double getHeight()
    double getCenter()
    double getFWHM()
    double getSigma()
    bool checkMaximalRTSpan(double max_rt_span)
    bool checkMinimalRTSpan(libcpp_pair[ double, double ] & rt_bounds, double\
       min_rt_span)
    double computeTheoretical(MassTrace & trace, Size k)
    double getArea()
    String getGnuplotFormula(MassTrace & trace, char function_name, double baseline,\
       double rt_shift)
    double getValue(double rt)
```

6.128 GoodDiffFilter

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/TRANSFORMERS/GoodDiffFilter.h>" namespace\
       "OpenMS":
```

```
  cdef cppclass GoodDiffFilter(FilterFunctor) :
    # wrap-inherits:
    #  FilterFunctor
    GoodDiffFilter()
    GoodDiffFilter(GoodDiffFilter)
    double apply(MSSpectrum[Peak1D] & )
    # POINTER # FilterFunctor * create()
    String getProductName()
```

6.129 Gradient

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/Gradient.h>" namespace "OpenMS":

  cdef cppclass Gradient:

    Gradient()
    Gradient(Gradient)  # wrap-ignore

    void addEluent(String eluent)
    void clearEluents()
    libcpp_vector[String] getEluents()

    void addTimepoint(Int timepoint)
    void clearTimepoints()
    libcpp_vector[Int] getTimepoints()

    void setPercentage(String eluent, Int timepoint, UInt percentage)
    UInt getPercentage(String eluent, Int timepoint)

    void clearPercentages()
    bool isValid()
```

6.130 GridBasedCluster

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/COMPARISON/CLUSTERING/GridBasedCluster.h>" namespace\
       "OpenMS":
```

```
  cdef cppclass GridBasedCluster "OpenMS::GridBasedCluster":

    GridBasedCluster(GridBasedCluster)  #wrap-ignore
    GridBasedCluster(DPosition2 centre,
      DBoundingBox2 bounding_box,
      libcpp_vector[ int ] point_indices,
      int property_A,
      libcpp_vector[ int ] properties_B)
    GridBasedCluster(DPosition2 centre,
       DBoundingBox2 bounding_box,
       libcpp_vector[ int ] point_indices)

    DPosition2 getCentre()
    DBoundingBox2 getBoundingBox()
    libcpp_vector[ int ] getPoints()
    int getPropertyA()
    libcpp_vector[ int ] getPropertiesB()
```

6.131 HPLC

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/HPLC.h>" namespace "OpenMS":

  cdef cppclass HPLC:

    HPLC()
    HPLC(HPLC)  # wrap-ignore

    String getInstrument()
    void setInstrument(String instrument)

    String getColumn()
    void setColumn(String column)

    Int getTemperature()
    void setTemperature(Int temperature)

    UInt getPressure()
    void setPressure(UInt pressure)

    UInt getFlux()
    void setFlux(UInt flux)

    String getComment()
```

```
    void setComment(String comment)

    Gradient getGradient()
    void setGradient(Gradient gradient)
```

6.132 HiddenMarkovModel

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/ID/HiddenMarkovModel.h>" namespace "OpenMS":

  cdef cppclass HiddenMarkovModel "OpenMS::HiddenMarkovModel":
    HiddenMarkovModel()
    HiddenMarkovModel(HiddenMarkovModel)
    void writeGraphMLFile(String & filename)
    double getTransitionProbability(String & s1, String & s2)
    void setTransitionProbability(String & s1, String & s2, double prob)
    Size getNumberOfStates()
    void addNewState(HMMState * state)
    void addNewState(String & name)
    void addSynonymTransition(String & name1, String & name2, String & synonym1,\
       String & synonym2)
    void evaluate()
    void train()
    void setInitialTransitionProbability(String & state, double prob)
    void clearInitialTransitionProbabilities()
    void setTrainingEmissionProbability(String & state, double prob)
    void clearTrainingEmissionProbabilities()
    void enableTransition(String & s1, String & s2)
    void disableTransition(String & s1, String & s2)
    void disableTransitions()
    void dump()
    void forwardDump()
    void estimateUntrainedTransitions()
    HMMState * getState(String & name)
    void clear()
    void setPseudoCounts(double pseudo_counts)
    double getPseudoCounts()
    void setVariableModifications(StringList & modifications)

cdef extern from "<OpenMS/ANALYSIS/ID/HiddenMarkovModel.h>" namespace "OpenMS":

  cdef cppclass HMMState "OpenMS::HMMState":
    HMMState()
    HMMState(HMMState)
```

```
    HMMState(String & name, bool hidden)

    void setName(String & name)
    String  getName()
    void setHidden(bool hidden)
    bool isHidden()
    void addPredecessorState(HMMState * state)
    void deletePredecessorState(HMMState * state)
    void addSuccessorState(HMMState * state)
    void deleteSuccessorState(HMMState * state)
    libcpp_set[ HMMState * ]  getPredecessorStates()
    libcpp_set[ HMMState * ]  getSuccessorStates()
```

6.133 HyperScore

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/RNPXL/HyperScore.h>" namespace "OpenMS":

  cdef cppclass HyperScore "OpenMS::HyperScore":
    HyperScore()
    HyperScore(HyperScore)  #wrap-ignore

    double compute(double fragment_mass_tolerance,
             bool fragment_mass_tolerance_unit_ppm,
             MSSpectrum[Peak1D] & exp_spectrum, MSSpectrum[RichPeak1D] &\
       theo_spectrum)
```

6.134 IBSpectraFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/IBSpectraFile.h>" namespace "OpenMS":

  cdef cppclass IBSpectraFile "OpenMS::IBSpectraFile":
    IBSpectraFile()
    IBSpectraFile(IBSpectraFile)
    void store(String & filename, ConsensusMap & cm)
```

6.135 IDDecoyProbability

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/ID/IDDecoyProbability.h>" namespace "OpenMS":

  cdef cppclass IDDecoyProbability(DefaultParamHandler):
    # wrap-inherits:
    #  DefaultParamHandler

    IDDecoyProbability()
    IDDecoyProbability(IDDecoyProbability)

    void apply(libcpp_vector[PeptideIdentification] & prob_ids,\
       libcpp_vector[PeptideIdentification] & fwd_ids,\
       libcpp_vector[PeptideIdentification] & rev_ids)
    void apply(libcpp_vector[PeptideIdentification] & ids)
```

6.136 IDFilter

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/ID/IDFilter.h>" namespace "OpenMS":

  cdef cppclass IDFilter:

    IDFilter()
    IDFilter(IDFilter)    # wrap-ignore

    Size countHits(libcpp_vector[PeptideIdentification] identifications)
    Size countHits(libcpp_vector[ProteinIdentification] identifications)

    bool getBestHit(libcpp_vector[PeptideIdentification] identifications, bool\
       assume_sorted, PeptideHit& best_hit)
    bool getBestHit(libcpp_vector[ProteinIdentification] identifications, bool\
       assume_sorted, ProteinHit& best_hit)

    void extractPeptideSequences(libcpp_vector[PeptideIdentification]& peptides,\
       libcpp_set[String]& sequences, bool ignore_mods)

    void updateHitRanks(libcpp_vector[PeptideIdentification]& identifications)
    void updateHitRanks(libcpp_vector[ProteinIdentification]& identifications)

    void removeUnreferencedProteins(libcpp_vector[ProteinIdentification]& proteins,\
```

```
   libcpp_vector[PeptideIdentification]& peptides)

void updateProteinReferences(libcpp_vector[PeptideIdentification]& peptides,\
   libcpp_vector[ProteinIdentification]& proteins, bool\
   remove_peptides_without_reference)

bool updateProteinGroups(libcpp_vector[ProteinGroup]& groups,\
   libcpp_vector[ProteinHit]& hits)

void removeEmptyIdentifications(libcpp_vector[PeptideIdentification]& ids)
void removeEmptyIdentifications(libcpp_vector[ProteinIdentification]& ids)

void filterHitsByScore(libcpp_vector[PeptideIdentification]& ids, double\
   threshold_score)
void filterHitsByScore(libcpp_vector[ProteinIdentification]& ids, double\
   threshold_score)

void filterHitsBySignificance(libcpp_vector[PeptideIdentification]& ids, double\
   threshold_fraction)
void filterHitsBySignificance(libcpp_vector[ProteinIdentification]& ids, double\
   threshold_fraction)

void keepNBestHits(libcpp_vector[PeptideIdentification]& ids, Size n)
void keepNBestHits(libcpp_vector[ProteinIdentification]& ids, Size n)

void filterHitsByRank(libcpp_vector[PeptideIdentification]& ids, Size min_rank,\
   Size max_rank)
void filterHitsByRank(libcpp_vector[ProteinIdentification]& ids, Size min_rank,\
   Size max_rank)

void removeDecoyHits(libcpp_vector[PeptideIdentification]& ids)
void removeDecoyHits(libcpp_vector[ProteinIdentification]& ids)

void removeHitsMatchingProteins(libcpp_vector[PeptideIdentification]& ids,\
   libcpp_set[String] accessions)
void removeHitsMatchingProteins(libcpp_vector[ProteinIdentification]& ids,\
   libcpp_set[String] accessions)

void keepHitsMatchingProteins(libcpp_vector[PeptideIdentification]& ids,\
   libcpp_set[String] accessions)
void keepHitsMatchingProteins(libcpp_vector[ProteinIdentification]& ids,\
   libcpp_set[String] accessions)

void keepBestPeptideHits(libcpp_vector[PeptideIdentification]& peptides, bool\
   strict)

void filterPeptidesByLength(libcpp_vector[PeptideIdentification]& peptides, Size\
   min_length, Size max_length)
```

```
void filterPeptidesByCharge(libcpp_vector[PeptideIdentification]& peptides, Size\
   min_charge, Size max_charge)

void filterPeptidesByRT(libcpp_vector[PeptideIdentification]& peptides, Size\
   min_rt, Size max_rt)

void filterPeptidesByMZ(libcpp_vector[PeptideIdentification]& peptides, Size\
   min_mz, Size max_mz)

void filterPeptidesByMZError(libcpp_vector[PeptideIdentification]& peptides,\
   double mass_error, bool unit_ppm)

void filterPeptidesByRTPredictPValue(libcpp_vector[PeptideIdentification]&\
   peptides, String& metavalue_key, double threshold)

void\
   removePeptidesWithMatchingModifications(libcpp_vector[PeptideIdentification]&\
   peptides, libcpp_set[String]& modifications)

void keepPeptidesWithMatchingModifications(libcpp_vector[PeptideIdentification]&\
   peptides, libcpp_set[String]& modifications)

void removePeptidesWithMatchingSequences(libcpp_vector[PeptideIdentification]&\
   peptides, libcpp_vector[PeptideIdentification]& bad_peptides, bool ignore_mods)

void keepPeptidesWithMatchingSequences(libcpp_vector[PeptideIdentification]&\
   peptides, libcpp_vector[PeptideIdentification]& bad_peptides, bool ignore_mods)

void keepUniquePeptidesPerProtein(libcpp_vector[PeptideIdentification]&\
   peptides)

void removeDuplicatePeptideHits(libcpp_vector[PeptideIdentification]& peptides)

void filterHitsByScore(MSExperiment[Peak1D, ChromatogramPeak]& experiment,\
   double peptide_threshold_score, double protein_threshold_score)

void filterHitsBySignificance(MSExperiment[Peak1D, ChromatogramPeak]&\
   experiment, double peptide_threshold_fraction, double protein_threshold_fraction)

void keepNBestHits(MSExperiment[Peak1D, ChromatogramPeak]& experiment, Size n)

void keepHitsMatchingProteins(MSExperiment[Peak1D, ChromatogramPeak]&\
   experiment, libcpp_vector[FASTAEntry]& proteins)
```

6.137 IDMapper

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/ID/IDMapper.h>" namespace "OpenMS":

  cdef cppclass IDMapper(DefaultParamHandler):
    # wrap-inherits:
    #  DefaultParamHandler

    IDMapper()
    IDMapper(IDMapper)

    void annotate(ConsensusMap & map_,
            libcpp_vector[PeptideIdentification] & ids,
            libcpp_vector[ProteinIdentification] & protein_ids,
            bool measure_from_subelements)

    void annotate(FeatureMap & map_,
            libcpp_vector[PeptideIdentification] & ids,
            libcpp_vector[ProteinIdentification] & protein_ids,
            bool use_centroid_rt,
            bool use_centroid_mz)

    void annotate(MSExperiment[Peak1D, ChromatogramPeak] & map_,
            libcpp_vector[PeptideIdentification] & ids,
            libcpp_vector[ProteinIdentification] & protein_ids)

cdef extern from "<OpenMS/ANALYSIS/ID/IDMapper.h>" namespace "OpenMS::IDMapper":
  cdef enum Measure:
    MEASURE_PPM = 0,
    MEASURE_DA
```

6.138 IDRipper

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/ID/IDRipper.h>" namespace "OpenMS":

  cdef cppclass IDRipper(DefaultParamHandler):
    # wrap-inherits:
    #  DefaultParamHandler

    IDRipper()
    IDRipper(IDRipper)    # wrap-ignore
```

```
    void rip(
      libcpp_map[String, libcpp_pair[ libcpp_vector[ProteinIdentification],
      libcpp_vector[PeptideIdentification]]] & ripped,
      libcpp_vector[ProteinIdentification] & proteins,
      libcpp_vector[PeptideIdentification] & peptides) # wrap-ignore
```

6.139 ILPDCWrapper

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/DECHARGING/ILPDCWrapper.h>" namespace "OpenMS":

  cdef cppclass ILPDCWrapper "OpenMS::ILPDCWrapper":
    ILPDCWrapper()
    ILPDCWrapper(ILPDCWrapper)  #wrap-ignore
    double compute(FeatureMap fm, libcpp_vector[ChargePair] & pairs, Size\
       verbose_level)
```

6.140 IMSAlphabet

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/MASSDECOMPOSITION/IMS/IMSAlphabet.h>" namespace\
       "OpenMS::ims::IMSAlphabet":

  ctypedef IMSElement element_type
  ctypedef libcpp_vector[element_type] container
  ctypedef libcpp_vector[mass_type] masses_type

cdef extern from "<OpenMS/CHEMISTRY/MASSDECOMPOSITION/IMS/IMSAlphabet.h>" namespace\
       "OpenMS::ims":

  cdef cppclass IMSAlphabet "OpenMS::ims::IMSAlphabet":
    IMSAlphabet()
    IMSAlphabet(IMSAlphabet)
    element_type  getElement(name_type & name)
    name_type  getName(size_type index)
    mass_type getMass(name_type & name)
    mass_type getMass(size_type index)
    masses_type getMasses(size_type isotope_index)
    masses_type getAverageMasses()
```

```
    bool hasName(name_type & name)
    void push_back(name_type & name, mass_type value)
    void push_back(element_type & element)
    void clear()
    void sortByNames()
    void sortByValues()
    void load(libcpp_string & fname)
    IMSAlphabet(libcpp_vector[IMSElement] & elements)
    size_type size()
    element_type  getElement(size_type index)
    void setElement(name_type & name, mass_type mass, bool forced)
    bool erase(name_type & name)
```

6.141 IMSAlphabetParser

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
#     IMSAlphabetParser(IMSAlphabetParser)  #wrap-ignore
#     void load(libcpp_string & fname)
#     # ContainerType getElements()
#     # void parse(InputSource & is_)
```

6.142 IMSAlphabetTextParser

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
#     # wrap-inherits:
#     #  IMSAlphabetParser
#     IMSAlphabetTextParser()
#     IMSAlphabetTextParser(IMSAlphabetTextParser)  #wrap-ignore
#     # ContainerType  getElements()
#     # NAMESPACE # void parse(std::istream & is_)
#
```

6.143 IMSDataConsumer

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/INTERFACES/IMSDataConsumer.h>" namespace\
       "OpenMS::Interfaces":
```

```
  cdef cppclass IMSDataConsumer[SpectrumType, ChromatogramType]:
    # wrap-ignore
    # ABSTRACT class

    void consumeSpectrum(SpectrumType &)

    void consumeChromatogram(ChromatogramType &)

    void setExpectedSize(Size, Size)

    void setExperimentalSettings(ExperimentalSettings &)
```

6.144 IMSElement

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/MASSDECOMPOSITION/IMS/IMSElement.h>" namespace\
       "OpenMS::ims::IMSElement":

  ctypedef libcpp_string name_type
  ctypedef IMSIsotopeDistribution isotopes_type

cdef extern from "<OpenMS/CHEMISTRY/MASSDECOMPOSITION/IMS/IMSElement.h>" namespace\
       "OpenMS::ims":

  cdef cppclass IMSElement "OpenMS::ims::IMSElement":
    IMSElement()
    IMSElement(IMSElement)
    IMSElement(name_type & name, isotopes_type & isotopes)
    IMSElement(name_type & name, mass_type mass)
    IMSElement(name_type & name, nominal_mass_type nominal_mass)
    name_type  getName()
    void setName(name_type & name)
    name_type  getSequence()
    void setSequence(name_type & sequence)
    nominal_mass_type getNominalMass()
    mass_type getMass(size_type index)
    mass_type getAverageMass()
    mass_type getIonMass(int electrons_number)
    IMSIsotopeDistribution  getIsotopeDistribution()
    void setIsotopeDistribution(IMSIsotopeDistribution & isotopes)
    bool operator==(IMSElement & element)
    bool operator!=(IMSElement & element)
```

6.145 IMSIsotopeDistribution

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/MASSDECOMPOSITION/IMS/IMSIsotopeDistribution.h>"\
       namespace "OpenMS::ims::IMSIsotopeDistribution":

  ctypedef signed int size_type
  ctypedef double mass_type
  ctypedef double abundance_type
  ctypedef unsigned int nominal_mass_type
  ctypedef libcpp_vector[mass_type] masses_container
  ctypedef IMSIsotopeDistribution_Peak peak_type
  ctypedef libcpp_vector[peak_type] peaks_container
  ctypedef libcpp_vector[abundance_type] abundances_container

cdef extern from "<OpenMS/CHEMISTRY/MASSDECOMPOSITION/IMS/IMSIsotopeDistribution.h>"\
       namespace "OpenMS::ims":

  cdef cppclass IMSIsotopeDistribution "OpenMS::ims::IMSIsotopeDistribution":
    IMSIsotopeDistribution()
    IMSIsotopeDistribution(IMSIsotopeDistribution)
    abundance_type ABUNDANCES_SUM_ERROR
    size_type SIZE
    IMSIsotopeDistribution(nominal_mass_type nominalMass)
    IMSIsotopeDistribution(mass_type mass)
    IMSIsotopeDistribution(libcpp_vector[IMSIsotopeDistribution_Peak] & peaks,\
       nominal_mass_type nominalMass)
    size_type size()
    bool operator==(IMSIsotopeDistribution & distribution)
    bool operator!=(IMSIsotopeDistribution & distribution)
    mass_type getMass(size_type i)
    abundance_type getAbundance(size_type i)
    mass_type getAverageMass()
    nominal_mass_type getNominalMass()
    void setNominalMass(nominal_mass_type nominalMass)
    masses_container getMasses()
    libcpp_vector[abundance_type] getAbundances()
    void normalize()
    bool empty()

cdef extern from "<OpenMS/CHEMISTRY/MASSDECOMPOSITION/IMS/IMSIsotopeDistribution.h>"\
       namespace "OpenMS::ims::IMSIsotopeDistribution":

  cdef cppclass IMSIsotopeDistribution_Peak\
       "OpenMS::ims::IMSIsotopeDistribution::Peak":
    IMSIsotopeDistribution_Peak(IMSIsotopeDistribution_Peak)  #wrap-ignore
```

```
    mass_type mass
    abundance_type abundance
    IMSIsotopeDistribution_Peak(mass_type mass, abundance_type abundance)
    bool operator==(IMSIsotopeDistribution_Peak & peak)
```

6.146 ISpectrumAccess

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from\
       "<OpenMS/ANALYSIS/OPENSWATH/OPENSWATHALGO/DATAACCESS/ISpectrumAccess.h>" namespace\
       "OpenSwath":

  cdef cppclass ISpectrumAccess:
    # wrap-ignore
    # ABSTRACT class

    ISpectrumAccess()
    ISpectrumAccess(ISpectrumAccess)

    shared_ptr[OSSpectrum] getSpectrumById(int id_)
    libcpp_vector[size_t] getSpectraByRT(double RT, double deltaRT)
    size_t getNrSpectra()
    shared_ptr[OSChromatogram] getChromatogramById(int id_)
    size_t getNrChromatograms()
    libcpp_string getChromatogramNativeID(int id_)
```

6.147 IdXMLFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/IdXMLFile.h>" namespace "OpenMS":

  cdef cppclass IdXMLFile:

    IdXMLFile()

    void load(String filename,
          libcpp_vector[ProteinIdentification] & protein_ids,
          libcpp_vector[PeptideIdentification] & peptide_ids,
          )
```

```
    void store(String filename,
          libcpp_vector[ProteinIdentification] & protein_ids,
          libcpp_vector[PeptideIdentification] & peptide_ids,
          String document_id)

    void store(String filename,
          libcpp_vector[ProteinIdentification] & protein_ids,
          libcpp_vector[PeptideIdentification] & peptide_ids,
          )
```

6.148 Identification

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/Identification.h>" namespace "OpenMS":

  cdef cppclass Identification(MetaInfoInterface):
    # wrap-inherits:
    #   MetaInfoInterface

    Identification()
    Identification(Identification)  # wrap-ignore

    void setCreationDate(DateTime date)
    DateTime getCreationDate()

    void setSpectrumIdentifications(libcpp_vector[SpectrumIdentification] & ids)

    void addSpectrumIdentification(SpectrumIdentification & id)

    libcpp_vector[SpectrumIdentification] getSpectrumIdentifications()

    void getKeys(libcpp_vector[String] & keys)
    void getKeys(libcpp_vector[unsigned int] & keys)  # wrap-as:getKeysAsIntegers
    DataValue getMetaValue(unsigned int)
    DataValue getMetaValue(String)
    void setMetaValue(unsigned int, DataValue)
    void setMetaValue(String, DataValue)
    bool metaValueExists(String)
    bool metaValueExists(unsigned int)
    void removeMetaValue(String)
    void removeMetaValue(unsigned int)
```

6.149 IdentificationHit

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/IdentificationHit.h>" namespace "OpenMS":

  cdef cppclass IdentificationHit(MetaInfoInterface):
    # wrap-inherits:
    #   MetaInfoInterface

    IdentificationHit()
    IdentificationHit(IdentificationHit)  # wrap-ignore

    void setId(String id)

    String getId()

    void setCharge(Int charge)

    Int getCharge()

    void setCalculatedMassToCharge(double mz)

    double getCalculatedMassToCharge()

    void setExperimentalMassToCharge(double mz)

    double getExperimentalMassToCharge()

    void setName(String name)

    String getName()

    void setPassThreshold(bool)

    bool getPassThreshold()

    void setRank(Int rank)

    Int getRank()

    void getKeys(libcpp_vector[String] & keys)
    void getKeys(libcpp_vector[unsigned int] & keys)  # wrap-as:getKeysAsIntegers
    DataValue getMetaValue(unsigned int)
    DataValue getMetaValue(String)
    void setMetaValue(unsigned int, DataValue)
    void setMetaValue(String, DataValue)
```

```
    bool metaValueExists(String)
    bool metaValueExists(unsigned int)
    void removeMetaValue(String)
    void removeMetaValue(unsigned int)
```

6.150 IncludeExcludeTarget

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/TARGETED/IncludeExcludeTarget.h>" namespace\
       "OpenMS":

  cdef cppclass IncludeExcludeTarget :
    IncludeExcludeTarget()
    IncludeExcludeTarget(IncludeExcludeTarget)
    void setName(String & name)
    String  getName()
    void setPeptideRef(String & peptide_ref)
    String  getPeptideRef()
    void setCompoundRef(String & compound_ref)
    String  getCompoundRef()
    void setPrecursorMZ(double mz)
    double getPrecursorMZ()
    void setPrecursorCVTermList(CVTermList & list_)
    void addPrecursorCVTerm(CVTerm & cv_term)
    CVTermList  getPrecursorCVTermList()
    void setProductMZ(double mz)
    double getProductMZ()
    void setProductCVTermList(CVTermList & list_)
    void addProductCVTerm(CVTerm & cv_term)
    CVTermList  getProductCVTermList()
    void setInterpretations(libcpp_vector[ CVTermList ] & interpretations)
    libcpp_vector[ CVTermList ]  getInterpretations()
    void addInterpretation(CVTermList & interpretation)
    void setConfigurations(libcpp_vector[ Configuration ] & configuration)
    libcpp_vector[ Configuration ]  getConfigurations()
    void addConfiguration(Configuration & configuration)
    void setPrediction(CVTermList & prediction)
    void addPredictionTerm(CVTerm & prediction)
    CVTermList  getPrediction()
    void setRetentionTime(RetentionTime rt)
    RetentionTime  getRetentionTime()
    bool operator==(IncludeExcludeTarget & rhs)
    bool operator!=(IncludeExcludeTarget & rhs)

    void setCVTerms(libcpp_vector[CVTerm] & terms)
```

```
    void replaceCVTerm(CVTerm & term)

    void replaceCVTerms(libcpp_vector[CVTerm] cv_terms,
              String accession
               )

    void replaceCVTerms(Map[String, libcpp_vector[CVTerm] ] cv_term_map
               )

    Map[String, libcpp_vector[CVTerm] ] getCVTerms()
    void addCVTerm(CVTerm & term)

    bool hasCVTerm(String accession)
    bool empty()
```

6.151 InclusionExclusionList

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/TARGETED/InclusionExclusionList.h>" namespace\
       "OpenMS":

  cdef cppclass InclusionExclusionList(DefaultParamHandler) :
    # wrap-inherits:
    #  DefaultParamHandler
    InclusionExclusionList()
    InclusionExclusionList(InclusionExclusionList)  #wrap-ignore
    void writeTargets(libcpp_vector[ FASTAEntry ] & fasta_entries, String &\
       out_path, IntList & charges, String rt_model_path)
    void writeTargets(FeatureMap & map_, String & out_path)
    void writeTargets(libcpp_vector[ PeptideIdentification ] & pep_ids, String &\
       out_path, IntList & charges)
```

6.152 IndexedMzMLDecoder

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/HANDLERS/IndexedMzMLDecoder.h>" namespace\
       "OpenMS":

  cdef cppclass IndexedMzMLDecoder:
    IndexedMzMLDecoder()
    IndexedMzMLDecoder(IndexedMzMLDecoder)
```

```
    int parseOffsets(String in_, int indexoffset,
        libcpp_vector[ libcpp_pair[ libcpp_string, streampos] ]& spectra_offsets,
        libcpp_vector[ libcpp_pair[ libcpp_string, streampos] ]&\
       chromatograms_offsets)  #wrap-ignore
    streampos findIndexListOffset(String in_, int buffersize)
```

6.153 IndexedMzMLFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/IndexedMzMLFile.h>" namespace "OpenMS":

  cdef cppclass IndexedMzMLFile "OpenMS::IndexedMzMLFile":
    IndexedMzMLFile()
    IndexedMzMLFile(IndexedMzMLFile)
    IndexedMzMLFile(String filename)
    void openFile(String filename)
    bool getParsingSuccess()
    size_t getNrSpectra()
    size_t getNrChromatograms()
    shared_ptr[Spectrum] getSpectrumById(int id_)
    shared_ptr[Chromatogram] getChromatogramById(int id_)
    void setSkipXMLChecks(bool skip)
```

6.154 IndexedMzMLFileLoader

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/IndexedMzMLFileLoader.h>" namespace "OpenMS":

  cdef cppclass IndexedMzMLFileLoader:

    IndexedMzMLFileLoader()

    bool load(String, OnDiscMSExperiment[Peak1D, ChromatogramPeak] &) nogil except+
    void store(String, OnDiscMSExperiment[Peak1D, ChromatogramPeak] &) nogil\
       except+
    void store(String, MSExperiment[Peak1D, ChromatogramPeak] &) nogil except+

    PeakFileOptions getOptions()
    void setOptions(PeakFileOptions)
```

6.155 InspectInfile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/InspectInfile.h>" namespace "OpenMS":

  cdef cppclass InspectInfile "OpenMS::InspectInfile":

    InspectInfile()
    InspectInfile(InspectInfile)

    bool operator==(InspectInfile & inspect_infile)
    void store(String & filename)
    void handlePTMs(String & modification_line, String & modifications_filename,\
       bool monoisotopic)
    String  getSpectra()
    void setSpectra(String & spectra)
    String  getDb()
    void setDb(String & db)
    String  getEnzyme()
    void setEnzyme(String & enzyme)
    Int getModificationsPerPeptide()
    void setModificationsPerPeptide(Int modifications_per_peptide)
    UInt getBlind()
    void setBlind(UInt blind)
    float getMaxPTMsize()
    void setMaxPTMsize(float maxptmsize)
    float getPrecursorMassTolerance()
    void setPrecursorMassTolerance(float precursor_mass_tolerance)
    float getPeakMassTolerance()
    void setPeakMassTolerance(float peak_mass_tolerance)
    UInt getMulticharge()
    void setMulticharge(UInt multicharge)
    String  getInstrument()
    void setInstrument(String & instrument)
    Int getTagCount()
    void setTagCount(Int TagCount)

    libcpp_map[ String, libcpp_vector[ String ] ]  getModifications()  #\
       wrap-ignore
```

6.156 InspectOutfile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/InspectOutfile.h>" namespace "OpenMS":

  cdef cppclass InspectOutfile "OpenMS::InspectOutfile":
    InspectOutfile()
    InspectOutfile(InspectOutfile)

    bool operator==(InspectOutfile & inspect_outfile)

    libcpp_vector[ size_t ] load(String & result_filename,
                  libcpp_vector[ PeptideIdentification ] &\
       peptide_identifications,
                  ProteinIdentification & protein_identification, double
                  p_value_threshold, String & database_filename)

    libcpp_vector[ size_t ] getWantedRecords(String & result_filename,
                         double p_value_threshold)

    void compressTrieDB(String & database_filename, String &
              index_filename, libcpp_vector[ size_t ] &
              wanted_records, String & snd_database_filename,
              String & snd_index_filename, bool append)

    void generateTrieDB(String & source_database_filename, String &
              database_filename, String & index_filename, bool
              append, String species)

    void getACAndACType(String line, String & accession, String & accession_type)

    void getPrecursorRTandMZ(
      libcpp_vector[ libcpp_pair[ String, libcpp_vector[ libcpp_pair[ size_t, size_t\
       ] ] ] ] & files_and_peptide_identification_with_scan_number,
      libcpp_vector[ PeptideIdentification ] & ids)  # wrap-ignore

    void getLabels(String & source_database_filename, String & ac_label,
             String & sequence_start_label, String & sequence_end_label,
             String & comment_label, String & species_label)

    libcpp_vector[ size_t ] getSequences(String & database_filename,
                      libcpp_map[ size_t, size_t ] & wanted_records,
                      libcpp_vector[ String ] & sequences)

    void getExperiment(MSExperiment[ Peak1D, ChromatogramPeak ] & exp, String &\
       type_, String & in_filename)

    bool getSearchEngineAndVersion(String & cmd_output, ProteinIdentification &\
       protein_identification)

    void readOutHeader(String & filename, String & header_line, Int &\
```

```
      spectrum_file_column, Int & scan_column, Int & peptide_column, Int &\
      protein_column, Int & charge_column, Int & MQ_score_column, Int & p_value_column,\
      Int & record_number_column, Int & DB_file_pos_column, Int & spec_file_pos_column,\
      Size & number_of_columns)
```

6.157 Instrument

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/Instrument.h>" namespace "OpenMS":

  cdef cppclass Instrument(MetaInfoInterface):
    # wrap-inherits:
    #  MetaInfoInterface

    Instrument()
    Instrument(Instrument)  # wrap-ignore

    String getName()
    void setName(String name)

    String getVendor()
    void setVendor(String vendor)

    String getModel()
    void setModel(String model)

    String getCustomizations()
    void setCustomizations(String customizations)

    libcpp_vector[IonSource] getIonSources()
    void setIonSources(libcpp_vector[IonSource] ion_sources)

    libcpp_vector[MassAnalyzer] getMassAnalyzers()
    void setMassAnalyzers(libcpp_vector[MassAnalyzer] mass_analyzers)

    libcpp_vector[IonDetector] getIonDetectors()
    void setIonDetectors(libcpp_vector[IonDetector] ion_detectors)

    Software getSoftware()
    void setSoftware(Software software)

    IonOpticsType getIonOptics()
    void setIonOptics(IonOpticsType ion_optics)
```

```
cdef extern from "<OpenMS/METADATA/Instrument.h>" namespace "OpenMS::Instrument":

  cdef enum IonOpticsType:

    UNKNOWN,                      #< unknown
    MAGNETIC_DEFLECTION,            #< magnetic deflection
    DELAYED_EXTRACTION,             #< delayed extraction
    COLLISION_QUADRUPOLE,         #< collision quadrupole
    SELECTED_ION_FLOW_TUBE,         #< selected ion flow tube
    TIME_LAG_FOCUSING,            #< time lag focusing
    REFLECTRON,                     #< reflectron
    EINZEL_LENS,                    #< einzel lens
    FIRST_STABILITY_REGION,         #< first stability region
    FRINGING_FIELD,                 #< fringing field
    KINETIC_ENERGY_ANALYZER,        #< kinetic energy analyzer
    STATIC_FIELD,                   #< static field
    SIZE_OF_IONOPTICSTYPE
```

6.158 InstrumentSettings

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/InstrumentSettings.h>" namespace "OpenMS":

  cdef cppclass InstrumentSettings(MetaInfoInterface):
    # wrap-inherits:
    #  MetaInfoInterface

    InstrumentSettings()
    InstrumentSettings(InstrumentSettings)

    Polarity getPolarity()
    void setPolarity(Polarity)

    ScanMode getScanMode()
    void setScanMode(ScanMode scan_mode)
    bool getZoomScan()
    void setZoomScan(bool zoom_scan)
    libcpp_vector[ ScanWindow ]  getScanWindows()
    void setScanWindows(libcpp_vector[ ScanWindow ] scan_windows)

    void getKeys(libcpp_vector[String] & keys)
    void getKeys(libcpp_vector[unsigned int] & keys)  # wrap-as:getKeysAsIntegers
    DataValue getMetaValue(unsigned int)
    DataValue getMetaValue(String)
```

```
    void setMetaValue(unsigned int, DataValue)
    void setMetaValue(String, DataValue)
    bool metaValueExists(String)
    bool metaValueExists(unsigned int)
    void removeMetaValue(String)
    void removeMetaValue(unsigned int)

cdef extern from "<OpenMS/METADATA/InstrumentSettings.h>" namespace\
       "OpenMS::InstrumentSettings":

  cdef enum ScanMode:
    UNKNOWN,            #< Unknown scan method
    MASSSPECTRUM,     #< general spectrum type
    MS1SPECTRUM,        #< full scan mass spectrum, is a "mass spectrum" @n\
       Synonyms: 'full spectrum', 'Q1 spectrum', 'Q3 spectrum', 'Single-Stage Mass\
       Spectrometry'
    MSNSPECTRUM,    #< MS2+ mass spectrum, is a "mass spectrum"
    SIM,                #< Selected ion monitoring scan @n Synonyms: 'Multiple ion\
       monitoring scan', 'SIM scan', 'MIM scan'
    SRM,                #< Selected reaction monitoring scan @n Synonyms: 'Multiple\
       reaction monitoring scan', 'SRM scan', 'MRM scan'
    CRM,                #< Consecutive reaction monitoring scan @n Synonyms: 'CRM scan'
    CNG,                #< Constant neutral gain scan @n Synonyms: 'CNG scan'
    CNL,                #< Constant neutral loss scan @n Synonyms: 'CNG scan'
    PRECURSOR,        #< Precursor ion scan
    EMC,                #< Enhanced multiply charged scan
    TDF,                #< Time-delayed fragmentation scan
    EMR,                #< Electromagnetic radiation scan @n Synonyms: 'EMR\
       spectrum'
    EMISSION,           #< Emission scan
    ABSORBTION,         #< Absorbtion scan
    SIZE_OF_SCANMODE
```

6.159 IntList

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
ctypedef libcpp_vector[ int ] IntList
```

6.160 IntegerMassDecomposer

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/MASSDECOMPOSITION/IMS/IntegerMassDecomposer.h>"\
       namespace "OpenMS::ims":

  cdef cppclass IntegerMassDecomposer[ValueType,DecompositionValueType]:
    # wrap-ignore

    # wrap-instances:
    #   IntegerMassDecomposer := IntegerMassDecomposer[int, int]
    IntegerMassDecomposer()  # wrap-ignore
    IntegerMassDecomposer(IntegerMassDecomposer)  #wrap-ignore
    IntegerMassDecomposer(IMSWeights & alphabet)
    bool exist(ValueType mass)
```

6.161 IntensityBalanceFilter

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/TRANSFORMERS/IntensityBalanceFilter.h>"\
       namespace "OpenMS":

  cdef cppclass IntensityBalanceFilter(FilterFunctor) :
    # wrap-inherits:
    #  FilterFunctor
    IntensityBalanceFilter()
    IntensityBalanceFilter(IntensityBalanceFilter)
    double apply(MSSpectrum[Peak1D] & )
    # POINTER # FilterFunctor * create()
    String getProductName()
```

6.162 InterfaceDataStructures

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/INTERFACES/DataStructures.h>" namespace\
       "OpenMS::Interfaces":

  cdef cppclass BinaryDataArray:
    # here we misuse wrap-instances for renaming the instance, wrap-as is not\
       supported for
    # classes, only for methods
    #
    # wrap-instances:
```

```
    #   _Interfaces_BinaryDataArray := BinaryDataArray
    BinaryDataArray()
    BinaryDataArray(BinaryDataArray)
    libcpp_vector[double] data

  ctypedef shared_ptr[BinaryDataArray] BinaryDataArrayPtr

  cdef cppclass Spectrum:
    # here we misuse wrap-instances for renaming the instance, wrap-as is not\
       supported for
    # classes, only for methods
    #
    # wrap-instances:
    #   _Interfaces_Spectrum := Spectrum
    Spectrum()
    Spectrum(Spectrum)
    BinaryDataArrayPtr getMZArray()  #wrap-ignore
    BinaryDataArrayPtr getIntensityArray()  #wrap-ignore
    void setMZArray(BinaryDataArrayPtr data)  #wrap-ignore
    void setIntensityArray(BinaryDataArrayPtr data)  #wrap-ignore

  ctypedef shared_ptr[Spectrum] SpectrumPtr

  cdef cppclass Chromatogram:
    # here we misuse wrap-instances for renaming the instance, wrap-as is not\
       supported for
    # classes, only for methods
    #
    # wrap-instances:
    #   _Interfaces_Chromatogram := Chromatogram
    Chromatogram()
    Chromatogram(Chromatogram)
    BinaryDataArrayPtr getTimeArray()  #wrap-ignore
    BinaryDataArrayPtr getIntensityArray()  #wrap-ignore
    void setTimeArray(BinaryDataArrayPtr data)  #wrap-ignore
    void setIntensityArray(BinaryDataArrayPtr data)  #wrap-ignore

  ctypedef shared_ptr[Chromatogram] ChromatogramPtr
```

6.163 InternalCalibration

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/CALIBRATION/InternalCalibration.h>" namespace\
       "OpenMS":
```

```
  cdef cppclass InternalCalibration(ProgressLogger):
    # wrap-inherits:
    #  ProgressLogger

    InternalCalibration()
    InternalCalibration(InternalCalibration)

    Size fillCalibrants(MSExperiment[Peak1D,ChromatogramPeak],
              libcpp_vector[InternalCalibration_LockMass],
              double tol_ppm,
              bool lock_require_mono,
              bool lock_require_iso,
              CalibrationData& failed_lock_masses,
              bool verbose)
    Size fillCalibrants(FeatureMap, double)
    Size fillCalibrants(libcpp_vector[PeptideIdentification], double)
    CalibrationData getCalibrationPoints()
    bool calibrate(MSExperiment[Peak1D,ChromatogramPeak],
             libcpp_vector[int],
             MZTrafoModel_MODELTYPE,
             double rt_chunk,
             bool use_RANSAC,
             double post_ppm_median,
             double post_ppm_MAD,
             String file_models,
             String file_models_plot,
             String file_residuals,
             String file_residuals_plot,
             String rscript_executable)

cdef extern from "<OpenMS/FILTERING/CALIBRATION/InternalCalibration.h>" namespace\
       "OpenMS::InternalCalibration":

  void applyTransformation(libcpp_vector[Precursor]& pcs,
               MZTrafoModel& trafo)  # wrap-attach:InternalCalibration
  void applyTransformation(MSSpectrum[Peak1D] & spec, IntList& target_mslvl,
               MZTrafoModel & trafo)  # wrap-attach:InternalCalibration
  void applyTransformation(MSExperiment[Peak1D, ChromatogramPeak] & exp,
               IntList& target_mslvl, MZTrafoModel& trafo)  #\
       wrap-attach:InternalCalibration

cdef extern from "<OpenMS/FILTERING/CALIBRATION/InternalCalibration.h>" namespace\
       "OpenMS::InternalCalibration":

  cdef cppclass InternalCalibration_LockMass\
       "OpenMS::InternalCalibration::LockMass":
    InternalCalibration_LockMass(InternalCalibration_LockMass)  # wrap-ignore
```

```
    InternalCalibration_LockMass(double mz_, int lvl_, int charge_)
    double mz # m/z of the lock mass (incl. adducts)
    unsigned int ms_level # MS level where it occurs
    int charge # charge of the ion (to find isotopes)
```

6.164 InterpolationModel

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
ctypedef double IntensityType
ctypedef DPosition1 PositionType
ctypedef double CoordinateType
cdef extern from "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/InterpolationModel.h>"\
       namespace "OpenMS":

  cdef cppclass InterpolationModel "OpenMS::InterpolationModel":
    InterpolationModel()
    InterpolationModel(InterpolationModel)

    double getIntensity(double coord)
    double getScalingFactor()
    void setOffset(double offset)
    double getCenter()
    void setSamples()
    void setInterpolationStep(double interpolation_step)
    void setScalingFactor(double scaling)

    LinearInterpolation[double,double] getInterpolation()
```

6.165 IonDetector

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/IonDetector.h>" namespace "OpenMS":

  cdef cppclass IonDetector(MetaInfoInterface):
    # wrap-inherits:
    #  MetaInfoInterface

    IonDetector()
    IonDetector(IonDetector)  # wrap-ignore

    Type_IonDetector getType()
```

```
    void setType(Type_IonDetector type_)

    AcquisitionMode getAcquisitionMode()
    void setAcquisitionMode(AcquisitionMode acquisition_mode)

    double getResolution()
    void setResolution(double resolution)

    double getADCSamplingFrequency()
    void setADCSamplingFrequency(double ADC_sampling_frequency)

    Int getOrder()
    void setOrder(Int order)

cdef extern from "<OpenMS/METADATA/IonDetector.h>" namespace "OpenMS::IonDetector":

    cdef enum Type_IonDetector "OpenMS::IonDetector::Type":
      # wrap-attach:
      #   IonDetector
      TYPENULL,                                  #< Unknown
      ELECTRONMULTIPLIER,                        #< Electron multiplier
      PHOTOMULTIPLIER,                           #< Photo multiplier
      FOCALPLANEARRAY,                           #< Focal plane array
      FARADAYCUP,                                #< Faraday cup
      CONVERSIONDYNODEELECTRONMULTIPLIER,        #< Conversion dynode electron\
       multiplier
      CONVERSIONDYNODEPHOTOMULTIPLIER,           #< Conversion dynode photo\
       multiplier
      MULTICOLLECTOR,                            #< Multi-collector
      CHANNELELECTRONMULTIPLIER,                #< Channel electron multiplier
      CHANNELTRON,                               #< channeltron
      DALYDETECTOR,                              #< daly detector
      MICROCHANNELPLATEDETECTOR,                #< microchannel plate detector
      ARRAYDETECTOR,                            #< array detector
      CONVERSIONDYNODE,                          #< conversion dynode
      DYNODE,                                    #< dynode
      FOCALPLANECOLLECTOR,                       #< focal plane collector
      IONTOPHOTONDETECTOR,                       #< ion-to-photon detector
      POINTCOLLECTOR,                            #< point collector
      POSTACCELERATIONDETECTOR,                  #< postacceleration detector
      PHOTODIODEARRAYDETECTOR,                   #< photodiode array detector
      INDUCTIVEDETECTOR,                        #< inductive detector
      ELECTRONMULTIPLIERTUBE,                    #< electron multiplier tube
      SIZE_OF_TYPE

    cdef enum AcquisitionMode:
      # wrap-attach:
      #   IonDetector
```

```
    ACQMODENULL,              #< Unknown
    PULSECOUNTING,          #< Pulse counting
    ADC,                      #< Analog-digital converter
    TDC,                      #< Time-digital converter
    TRANSIENTRECORDER,      #< Transient recorder
    SIZE_OF_ACQUISITIONMODE
```

6.166 IonSource

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/IonSource.h>" namespace "OpenMS":

  cdef cppclass IonSource(MetaInfoInterface):
    # wrap-inherits:
    #  MetaInfoInterface

    IonSource()
    IonSource(IonSource)  # wrap-ignore

    Polarity getPolarity()
    void setPolarity(Polarity polarity)

    InletType getInletType()
    void setInletType(InletType inlet_type)

    IonizationMethod getIonizationMethod()
    void setIonizationMethod(IonizationMethod ionization_type)

    Int getOrder()
    void setOrder(Int order)

cdef extern from "<OpenMS/METADATA/IonSource.h>" namespace "OpenMS::IonSource":

  cdef enum Polarity:
    # wrap-attach:
    #   IonSource
    POLNULL, POSITIVE, NEGATIVE, SIZE_OF_POLARITY

  cdef enum InletType:
    # wrap-attach:
    #   IonSource
    INLETNULL,                          #]Unknown
    DIRECT,                               #]Direct
    BATCH,                              #]Batch (e.g. in MALDI)
    CHROMATOGRAPHY,                       #]Chromatography (liquid)
```

```
  PARTICLEBEAM,                          #]Particle beam
  MEMBRANESEPARATOR,                    #]Membrane sparator
  OPENSPLIT,                            #]Open split
  JETSEPARATOR,                          #]Jet separator
  SEPTUM,                                #]Septum
  RESERVOIR,                            #]Reservoir
  MOVINGBELT,                            #]Moving belt
  MOVINGWIRE,                            #]Moving wire
  FLOWINJECTIONANALYSIS,                #]Flow injection analysis
  ELECTROSPRAYINLET,                    #]Electro spray
  THERMOSPRAYINLET,                      #]Thermo spray
  INFUSION,                              #]Infusion
  CONTINUOUSFLOWFASTATOMBOMBARDMENT,    #]Continuous flow fast atom bombardment
  INDUCTIVELYCOUPLEDPLASMA,              #]Inductively coupled plasma
  MEMBRANE,                              #]Membrane inlet
  NANOSPRAY,                            #]Nanospray inlet
  SIZE_OF_INLETTYPE

cdef enum IonizationMethod:
  # wrap-attach:
  #   IonSource
  IONMETHODNULL,    #]Unknown
  ESI,               #]electrospray ionisation
  EI,                #]electron ionization
  CI,                #]chemical ionisation
  FAB,               #]fast atom bombardment
  TSP,               #]thermospray
  LD,                #]laser desorption
  FD,                #]field desorption
  FI,                #]flame ionization
  PD,                #]plasma desorption
  SI,                #]secondary ion MS
  TI,                #]thermal ionization
  API,               #]atmospheric pressure ionisation
  ISI,               #<
  CID,               #]collsion induced decomposition
  CAD,               #]collsion activated decomposition
  HN,                #<
  APCI,              #]atmospheric pressure chemical ionization
  APPI,              #]atmospheric pressure photo ionization
  ICP,               #]inductively coupled plasma
  NESI,              #]Nano electrospray ionization
  MESI,              #]Micro electrospray ionization
  SELDI,            #]Surface enhanced laser desorption ionization
  SEND,              #]Surface enhanced neat desorption
  FIB,               #]Fast ion bombardment
  MALDI,            #]Matrix-assisted laser desorption ionization
  MPI,               #]Multiphoton ionization
```

```
    DI,                 #]desorption ionization
    FA,                 #]flowing afterglow
    FII,                #]field ionization
    GD_MS,             #]glow discharge ionization
    NICI,               #]negative ion chemical ionization
    NRMS,               #]neutralization reionization mass spectrometry
    PI,                 #]photoionization
    PYMS,               #]pyrolysis mass spectrometry
    REMPI,             #]resonance enhanced multiphoton ionization
    AI,                 #]adiabatic ionization
    ASI,                #]associative ionization
    AD,                 #]autodetachment
    AUI,                #]autoionization
    CEI,                #]charge exchange ionization
    CHEMI,             #]chemi-ionization
    DISSI,             #]dissociative ionization
    LSI,                #]liquid secondary ionization
    PEI,                #]penning ionization
    SOI,                #]soft ionization
    SPI,                #]spark ionization
    SUI,                #]surface ionization
    VI,                 #]vertical ionization
    AP_MALDI,           #]atmospheric pressure matrix-assisted laser desorption\
       ionization
    SILI,               #]desorption/ionization on silicon
    SALDI,             #]surface-assisted laser desorption ionization
    SIZE_OF_IONIZATIONMETHOD
```

6.167 IsobaricChannelExtractor

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/QUANTITATION/IsobaricChannelExtractor.h>"\
       namespace "OpenMS":

  cdef cppclass IsobaricChannelExtractor(DefaultParamHandler) :
    # wrap-inherits:
    #  DefaultParamHandler

    IsobaricChannelExtractor(IsobaricChannelExtractor)

    # IsobaricChannelExtractor(IsobaricQuantitationMethod *quant_method) \
       #wrap-ignore
    IsobaricChannelExtractor(ItraqEightPlexQuantitationMethod *quant_method)
    IsobaricChannelExtractor(ItraqFourPlexQuantitationMethod *quant_method)
    IsobaricChannelExtractor(TMTSixPlexQuantitationMethod *quant_method)
```

```
    IsobaricChannelExtractor(TMTTenPlexQuantitationMethod *quant_method)

    void extractChannels(MSExperiment[ Peak1D, ChromatogramPeak ] & ms_exp_data,\
       ConsensusMap & consensus_map)
```

6.168 IsobaricChannelInformation

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/QUANTITATION/IsobaricQuantitationMethod.h>"\
       namespace "OpenMS::IsobaricQuantitationMethod":

  cdef cppclass IsobaricChannelInformation\
       "OpenMS::IsobaricQuantitationMethod::IsobaricChannelInformation":
    IsobaricChannelInformation(IsobaricChannelInformation)  #wrap-ignore
    String name
    Int id
    String description
    double center
    Int channel_id_minus_2
    Int channel_id_minus_1
    Int channel_id_plus_1
    Int channel_id_plus_2
    IsobaricChannelInformation(String name, Int id_, String description, double\
       center, Int channel_id_minus_2, Int channel_id_minus_1, Int channel_id_plus_1, Int\
       channel_id_plus_2)
```

6.169 IsobaricIsotopeCorrector

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/QUANTITATION/IsobaricIsotopeCorrector.h>"\
       namespace "OpenMS":

  cdef cppclass IsobaricIsotopeCorrector:

    IsobaricIsotopeCorrector()
    IsobaricIsotopeCorrector(IsobaricIsotopeCorrector)  # wrap-ignore

    #                            IsobaricQuantitationMethod * quant_method)  #\
       wrap-ignore

    IsobaricQuantifierStatistics correctIsotopicImpurities(ConsensusMap &\
```

```
      consensus_map_in,
                                ConsensusMap & consensus_map_out,
                                ItraqEightPlexQuantitationMethod * quant_method)
    IsobaricQuantifierStatistics correctIsotopicImpurities(ConsensusMap &\
      consensus_map_in,
                                ConsensusMap & consensus_map_out,
                                ItraqFourPlexQuantitationMethod * quant_method)
    IsobaricQuantifierStatistics correctIsotopicImpurities(ConsensusMap &\
      consensus_map_in,
                                ConsensusMap & consensus_map_out,
                                TMTSixPlexQuantitationMethod * quant_method)
    IsobaricQuantifierStatistics correctIsotopicImpurities(ConsensusMap &\
      consensus_map_in,
                                ConsensusMap & consensus_map_out,
                                TMTTenPlexQuantitationMethod * quant_method)
```

6.170 IsobaricNormalizer

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/QUANTITATION/IsobaricNormalizer.h>" namespace\
       "OpenMS":

  cdef cppclass IsobaricNormalizer "OpenMS::IsobaricNormalizer":
    IsobaricNormalizer(IsobaricNormalizer)

    IsobaricNormalizer(IsobaricQuantitationMethod *quant_method)  # wrap-ignore
    IsobaricNormalizer(ItraqFourPlexQuantitationMethod *quant_method)
    IsobaricNormalizer(ItraqEightPlexQuantitationMethod *quant_method)
    IsobaricNormalizer(TMTSixPlexQuantitationMethod *quant_method)
    IsobaricNormalizer(TMTTenPlexQuantitationMethod *quant_method)

    void normalize(ConsensusMap & consensus_map)
```

6.171 IsobaricQuantifier

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/QUANTITATION/IsobaricQuantifier.h>" namespace\
       "OpenMS":

  cdef cppclass IsobaricQuantifier(DefaultParamHandler) :
```

```
    # wrap-inherits:
    #  DefaultParamHandler
    IsobaricQuantifier(IsobaricQuantifier)

    IsobaricQuantifier(IsobaricQuantitationMethod *quant_method)   # wrap-ignore
    IsobaricQuantifier(ItraqFourPlexQuantitationMethod *quant_method)
    IsobaricQuantifier(ItraqEightPlexQuantitationMethod *quant_method)
    IsobaricQuantifier(TMTSixPlexQuantitationMethod *quant_method)
    IsobaricQuantifier(TMTTenPlexQuantitationMethod *quant_method)

    void quantify(ConsensusMap & consensus_map_in, ConsensusMap & consensus_map_out)
```

6.172 IsobaricQuantifierStatistics

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/QUANTITATION/IsobaricQuantifierStatistics.h>"\
       namespace "OpenMS":

  cdef cppclass IsobaricQuantifierStatistics\
       "OpenMS::IsobaricQuantifierStatistics":
    IsobaricQuantifierStatistics()
    IsobaricQuantifierStatistics(IsobaricQuantifierStatistics)
    Size channel_count
    Size iso_number_ms2_negative
    Size iso_number_reporter_negative
    Size iso_number_reporter_different
    double iso_solution_different_intensity
    double iso_total_intensity_negative
    Size number_ms2_total
    Size number_ms2_empty
    void reset()
```

6.173 IsobaricQuantitationMethod

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/QUANTITATION/IsobaricQuantitationMethod.h>"\
       namespace "OpenMS":

  cdef cppclass IsobaricQuantitationMethod(DefaultParamHandler) :
    # wrap-ignore
    # ABSTRACT class
```

```
    # wrap-inherits:
    #  DefaultParamHandler
    IsobaricQuantitationMethod()
    IsobaricQuantitationMethod(IsobaricQuantitationMethod)  #wrap-ignore
    String getName()
    libcpp_vector[IsobaricChannelInformation]  getChannelInformation()
    Size getNumberOfChannels()
    Matrix[ double ] getIsotopeCorrectionMatrix()
    Size getReferenceChannel()
```

6.174 IsotopeCluster

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/DATASTRUCTURES/IsotopeCluster.h>" namespace "OpenMS":

  cdef cppclass IsotopeCluster "OpenMS::IsotopeCluster":
    IsotopeCluster()
    IsotopeCluster(IsotopeCluster)  #wrap-ignore
    ChargedIndexSet peaks
    libcpp_vector[ size_t ] scans

cdef extern from "<OpenMS/DATASTRUCTURES/IsotopeCluster.h>" namespace\
       "OpenMS::IsotopeCluster":

  cdef cppclass ChargedIndexSet "OpenMS::IsotopeCluster::ChargedIndexSet":
    ChargedIndexSet()
    ChargedIndexSet(ChargedIndexSet)  #wrap-ignore
    Int charge
```

6.175 IsotopeDiffFilter

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/TRANSFORMERS/IsotopeDiffFilter.h>" namespace\
       "OpenMS":

  cdef cppclass IsotopeDiffFilter(FilterFunctor) :
    # wrap-inherits:
    #  FilterFunctor
    IsotopeDiffFilter()
```

```
    IsotopeDiffFilter(IsotopeDiffFilter)
    double apply(MSSpectrum[Peak1D] & )
    # POINTER # FilterFunctor * create()
    String getProductName()
```

6.176 IsotopeDistribution

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/IsotopeDistribution.h>" namespace "OpenMS":

  cdef cppclass IsotopeDistribution:

    IsotopeDistribution()
    IsotopeDistribution(IsotopeDistribution)  # wrap-ignore

    void setMaxIsotope(Size max_isotope)

    Size getMaxIsotope()

    void set(libcpp_vector[ libcpp_pair[ size_t, double] ] & distribution)

    libcpp_vector[ libcpp_pair[ size_t, double] ] getContainer()

    Size getMax()

    Size getMin()

    Size size()

    void clear()

    void estimateFromPeptideWeight(double average_weight)

    void estimateFromRNAWeight(double average_weight)

    void estimateFromDNAWeight(double average_weight)

    void estimateFromWeightAndComp(double average_weight, double C, double H, double\
       N, double O, double S, double P)

    void renormalize()

    void trimRight(double cutoff)
```

```
    void trimLeft(double cutoff)
```

6.177 IsotopeDistributionCache

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/DATAREDUCTION/IsotopeDistributionCache.h>"\
       namespace "OpenMS":

  cdef cppclass IsotopeDistributionCache "OpenMS::IsotopeDistributionCache":
    IsotopeDistributionCache(IsotopeDistributionCache)  #wrap-ignore
    IsotopeDistributionCache(double max_mass, double mass_window_width, double\
       intensity_percentage, double intensity_percentage_optional)
    TheoreticalIsotopePattern  getIsotopeDistribution(double mass)
```

6.178 IsotopeFitter1D

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/IsotopeFitter1D.h>"\
       namespace "OpenMS":

  cdef cppclass IsotopeFitter1D "OpenMS::IsotopeFitter1D":
    IsotopeFitter1D()
    IsotopeFitter1D(IsotopeFitter1D)
    String getProductName()
```

6.179 IsotopeMarker

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/TRANSFORMERS/IsotopeMarker.h>" namespace\
       "OpenMS":

  cdef cppclass IsotopeMarker(PeakMarker) :
    # wrap-inherits:
    #  PeakMarker
    IsotopeMarker()
    IsotopeMarker(IsotopeMarker)
```

```
    void apply(libcpp_map[ double, bool ] & , MSSpectrum[Peak1D] & )
    PeakMarker * create()  # wrap-ignore
    # TODO
    #String getProductName()
```

6.180 IsotopeModel

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/IsotopeModel.h>" namespace\
       "OpenMS":

  cdef cppclass IsotopeModel "OpenMS::IsotopeModel":
    IsotopeModel()
    IsotopeModel(IsotopeModel)
    UInt getCharge()
    void setOffset(double offset)
    double getOffset()
    EmpiricalFormula getFormula()
    void setSamples(EmpiricalFormula &formula)
    double getCenter()
    IsotopeDistribution  getIsotopeDistribution()
    String getProductName()

cdef extern from "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/IsotopeModel.h>" namespace\
       "OpenMS::IsotopeModel":
  cdef enum Averagines "OpenMS::IsotopeModel::Averagines":
    #wrap-attach:
    #  IsotopeModel
    C
    H
    N
    O
    S
    AVERAGINE_NUM
```

6.181 IsotopeWavelet

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/IsotopeWavelet.h>" namespace\
       "OpenMS":
```

```
  cdef cppclass IsotopeWavelet "OpenMS::IsotopeWavelet":
    IsotopeWavelet(IsotopeWavelet)  #wrap-ignore
    # IsotopeWavelet * init(double max_m, UInt max_charge)
    # IsotopeWavelet * getInstance()
    void destroy()
    double getValueByMass(double t, double m, UInt z, Int mode)
    double getValueByLambda(double lambda_, double tz1)
    double getValueByLambdaExtrapol(double lambda_, double tz1)
    double getValueByLambdaExact(double lambda_, double tz1)
    UInt getMaxCharge()
    void setMaxCharge(UInt max_charge)
    double getTableSteps()
    double getInvTableSteps()
    void setTableSteps(double table_steps)
    double getLambdaL(double m)
    # IsotopeDistribution::ContainerType  getAveragine(double m, UInt *size)
    Size getGammaTableMaxIndex()
    Size getExpTableMaxIndex()
    float myPow(float a, float b)
    UInt getMzPeakCutOffAtMonoPos(double mass, UInt z)
    UInt getNumPeakCutOff(double mass, UInt z)
    UInt getNumPeakCutOff(double mz)
```

6.182 IsotopeWaveletTransform

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/IsotopeWaveletTransform.h>"\
       namespace "OpenMS":

  cdef cppclass IsotopeWaveletTransform[PeakT]:
    # wrap-instances:
    #   IsotopeWaveletTransform := IsotopeWaveletTransform[Peak1D]

    IsotopeWaveletTransform(IsotopeWaveletTransform)  #wrap-ignore
    IsotopeWaveletTransform(double min_mz, double max_mz, UInt max_charge, Size\
       max_scan_size, bool hr_data, String intenstype)
    void getTransform(MSSpectrum[ PeakT ] &c_trans, MSSpectrum[ PeakT ] &c_ref, UInt c)
    void getTransformHighRes(MSSpectrum[ PeakT ] &c_trans, MSSpectrum[ PeakT ]\
       &c_ref, UInt c)
    void identifyCharge(MSSpectrum[ PeakT ] &candidates, MSSpectrum[ PeakT ] &ref,\
       UInt scan_index, UInt c, double ampl_cutoff, bool check_PPMs)
    void initializeScan(MSSpectrum[ PeakT ] &c_ref, UInt c)
    void updateBoxStates(MSExperiment[ PeakT, ChromatogramPeak ] &map_, Size\
```

```
       scan_index, UInt RT_interleave, UInt RT_votes_cutoff, Int front_bound, Int\
       end_bound)
    # void mergeFeatures(IsotopeWaveletTransform[ PeakT ] *later_iwt, UInt\
       RT_interleave, UInt RT_votes_cutoff)
    FeatureMap mapSeeds2Features(MSExperiment[ PeakT, ChromatogramPeak] &map_, UInt\
       RT_votes_cutoff)
    ## std::multimap[ double, Box ] getClosedBoxes()
    ## double getLinearInterpolation(typename MSSpectrum[ PeakT ]::const_iterator\
       &left_iter, double mz_pos, typename MSSpectrum[ PeakT ]::const_iterator\
       &right_iter)
    double getLinearInterpolation(double mz_a, double intens_a, double mz_pos,\
       double mz_b, double intens_b)
    double getSigma()
    void setSigma(double sigma)
    void computeMinSpacing(MSSpectrum[ PeakT ] &c_ref)
    double getMinSpacing()
    Size getMaxScanSize()

#     TransSpectrum(TransSpectrum)  #wrap-ignore
#     # POINTER #  TransSpectrum(MSSpectrum[ PeakType ] * reference)
#     void destroy()
#     double getRT()
#     double getMZ(UInt i)
#     double getRefIntensity(UInt i)
#     double getTransIntensity(UInt i)
#     void setTransIntensity(UInt i, double intens)
#     Size size()
#     # POINTER # MSSpectrum[ PeakType ] * getRefSpectrum()
#     # POINTER # MSSpectrum[ PeakType ] * getRefSpectrum()
#     # NAMESPACE # MSSpectrum[ PeakType ]::const_iterator MZBegin(double mz)
#     # NAMESPACE # MSSpectrum[ PeakType ]::const_iterator MZEnd(double mz)
#     # NAMESPACE # MSSpectrum[ PeakType ]::const_iterator end()
#     # NAMESPACE # MSSpectrum[ PeakType ]::const_iterator begin()
#

cdef extern from "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/IsotopeWaveletTransform.h>"\
       namespace "OpenMS::IsotopeWaveletTransform":

  cdef cppclass BoxElement "OpenMS::IsotopeWaveletTransform::BoxElement":
    BoxElement(BoxElement)  #wrap-ignore
    double mz
    UInt c
    double score
    double intens
    double ref_intens
    double RT
    UInt RT_index
    UInt MZ_begin
```

```
    UInt MZ_end
```

6.183 ItraqChannelExractor

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/QUANTITATION/ItraqChannelExtractor.h>" namespace\
       "OpenMS":

  cdef cppclass ItraqChannelExtractor(ItraqConstants,DefaultParamHandler):
    # wrap-inherits:
    #  DefaultParamHandler
    #  ItraqConstants

    ItraqChannelExtractor()
    ItraqChannelExtractor(ItraqChannelExtractor)  #wrap-ignore
    ItraqChannelExtractor(Int itraq_type, Param param)

    void run(MSExperiment[Peak1D, ChromatogramPeak] & ms_exp, ConsensusMap &\
       map_out)
```

6.184 ItraqConstants

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/QUANTITATION/ItraqConstants.h>" namespace\
       "OpenMS":

  cdef cppclass ItraqConstants "OpenMS::ItraqConstants":
    ItraqConstants()
    ItraqConstants(ItraqConstants)  #wrap-ignore
    # Int CHANNEL_COUNT()
    # Int CHANNELS_FOURPLEX()
    # Int CHANNELS_EIGHTPLEX()
    # Int CHANNELS_TMT_SIXPLEX()
    # double ISOTOPECORRECTIONS_FOURPLEX()
    # double ISOTOPECORRECTIONS_EIGHTPLEX()
    # double ISOTOPECORRECTIONS_TMT_SIXPLEX()
    StringList getIsotopeMatrixAsStringList(int itraq_type,\
       libcpp_vector[Matrix[double] ] & isotope_corrections)
    void updateIsotopeMatrixFromStringList(int itraq_type, StringList & channels,\
       libcpp_vector[Matrix[double] ] & isotope_corrections)
```

```
    # void initChannelMap(int itraq_type, ChannelMapType & map_)
    # void updateChannelMap(StringList & active_channels, ChannelMapType & map_)
    Matrix[ double ] translateIsotopeMatrix(int & itraq_type,\
       libcpp_vector[Matrix[double] ] & isotope_corrections)

cdef extern from "<OpenMS/ANALYSIS/QUANTITATION/ItraqConstants.h>" namespace\
       "OpenMS::ItraqConstants":

  cdef enum ITRAQ_TYPES:

    FOURPLEX, EIGHTPLEX, TMT_SIXPLEX, SIZE_OF_ITRAQ_TYPES

  cdef cppclass ChannelInfo "OpenMS::ItraqConstants::ChannelInfo":
    ChannelInfo(ChannelInfo)  #wrap-ignore
    # TODO string variable
    # String description
    Int name
    Int id
    double center
    bool active
```

6.185 ItraqEightPlexQuantitationMethod

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/QUANTITATION/ItraqEightPlexQuantitationMethod.h>"\
       namespace "OpenMS":

  cdef cppclass ItraqEightPlexQuantitationMethod(IsobaricQuantitationMethod) :
    # wrap-inherits:
    #  IsobaricQuantitationMethod
    ItraqEightPlexQuantitationMethod()
    ItraqEightPlexQuantitationMethod(ItraqEightPlexQuantitationMethod)
```

6.186 ItraqFourPlexQuantitationMethod

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/QUANTITATION/ItraqFourPlexQuantitationMethod.h>"\
       namespace "OpenMS":

  cdef cppclass ItraqFourPlexQuantitationMethod(IsobaricQuantitationMethod) :
```

```
    # wrap-inherits:
    #  IsobaricQuantitationMethod
    ItraqFourPlexQuantitationMethod()
    ItraqFourPlexQuantitationMethod(ItraqFourPlexQuantitationMethod)
```

6.187 ItraqQuantifier

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/QUANTITATION/ItraqQuantifier.h>" namespace\
       "OpenMS":

  cdef cppclass ItraqQuantifier(ItraqConstants,DefaultParamHandler):
    # wrap-inherits:
    #  DefaultParamHandler
    #  ItraqConstants

    ItraqQuantifier()
    ItraqQuantifier(ItraqQuantifier)  #wrap-ignore
    ItraqQuantifier(Int itraq_type, Param param)

    void run(ConsensusMap & map_in, ConsensusMap & map_out)

    ItraqQuantifierStats getStats()

cdef extern from "<OpenMS/ANALYSIS/QUANTITATION/ItraqQuantifier.h>" namespace\
       "OpenMS::ItraqQuantifier":

  cdef cppclass ItraqQuantifierStats\
       "OpenMS::ItraqQuantifier::ItraqQuantifierStats":
    ItraqQuantifierStats()
    ItraqQuantifierStats(ItraqQuantifierStats)  #wrap-ignore
    Size channel_count
    Size iso_number_ms2_negative
    Size iso_number_reporter_negative
    Size iso_number_reporter_different
    double iso_solution_different_intensity
    double iso_total_intensity_negative
    Size number_ms2_total
    Size number_ms2_empty
    # TODO STL attribute
    libcpp_map[ size_t, size_t ] empty_channels
```

6.188 JavaInfo

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/SYSTEM/JavaInfo.h>" namespace "OpenMS":

  cdef cppclass JavaInfo:

    JavaInfo()
    JavaInfo(JavaInfo)  # wrap-ignore

    bool canRun(String java_executable)
```

6.189 KroenikFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/KroenikFile.h>" namespace "OpenMS":

  cdef cppclass KroenikFile:

    KroenikFile()

    void store(String filename, MSSpectrum[Peak1D] & spectrum)
    void load(String filename, FeatureMap & feature_map)
```

6.190 LPWrapper

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/DATASTRUCTURES/LPWrapper.h>" namespace "OpenMS":

  cdef cppclass LPWrapper "OpenMS::LPWrapper":
    LPWrapper()
    LPWrapper(LPWrapper)  #wrap-ignore
    Int addRow(libcpp_vector[ int ] row_indices, libcpp_vector[ double ] row_values,\
       String & name)
    Int addColumn()
    Int addColumn(libcpp_vector[ int ] column_indices, libcpp_vector[ double ]\
       column_values, String & name)
    Int addRow(libcpp_vector[ int ] & row_indices, libcpp_vector[ double ] &\
       row_values, String & name, double lower_bound, double upper_bound, LPWrapper_Type\
```

```
        type_)
      Int addColumn(libcpp_vector[ int ] & column_indices, libcpp_vector[ double ] &\
        column_values, String & name, double lower_bound, double upper_bound,\
        LPWrapper_Type type_)
      void deleteRow(Int index)
      void setColumnName(Int index, String & name)
      String getColumnName(Int index)
      String getRowName(Int index)
      Int getRowIndex(String & name)
      Int getColumnIndex(String & name)
      double getColumnUpperBound(Int index)
      double getColumnLowerBound(Int index)
      double getRowUpperBound(Int index)
      double getRowLowerBound(Int index)
      void setRowName(Int index, String & name)
      void setColumnBounds(Int index, double lower_bound, double upper_bound,\
        LPWrapper_Type type_)
      void setRowBounds(Int index, double lower_bound, double upper_bound,\
        LPWrapper_Type type_)
      void setColumnType(Int index, VariableType type_)
      VariableType getColumnType(Int index)
      void setObjective(Int index, double obj_value)
      double getObjective(Int index)
      void setObjectiveSense(Sense sense)
      Sense getObjectiveSense()
      Int getNumberOfColumns()
      Int getNumberOfRows()
      void setElement(Int row_index, Int column_index, double value)
      double getElement(Int row_index, Int column_index)
      void readProblem(String filename, String format_)
      void writeProblem(String & filename, WriteFormat format_)
      Int solve(SolverParam & solver_param, Size verbose_level)
      SolverStatus getStatus()
      double getObjectiveValue()
      double getColumnValue(Int index)
      Int getNumberOfNonZeroEntriesInRow(Int idx)
      void getMatrixRow(Int idx, libcpp_vector[ int ] & indexes)
      void setSolver(SOLVER s)
      SOLVER getSolver()

cdef extern from "<OpenMS/DATASTRUCTURES/LPWrapper.h>" namespace\
       "OpenMS::LPWrapper":

  cdef cppclass SolverParam "OpenMS::LPWrapper::SolverParam":
    #wrap-attach:
    #  LPWrapper
    SolverParam()
    SolverParam(SolverParam)  #wrap-ignore
```

```
    Int message_level
    Int branching_tech
    Int backtrack_tech
    Int preprocessing_tech
    bool enable_feas_pump_heuristic
    bool enable_gmi_cuts
    bool enable_mir_cuts
    bool enable_cov_cuts
    bool enable_clq_cuts
    double mip_gap
    Int time_limit
    Int output_freq
    Int output_delay
    bool enable_presolve
    bool enable_binarization

cdef extern from "<OpenMS/DATASTRUCTURES/LPWrapper.h>" namespace\
       "OpenMS::LPWrapper":
  cdef enum LPWrapper_Type "OpenMS::LPWrapper::Type":
    #wrap-attach:
    #  LPWrapper
    UNBOUNDED
    LOWER_BOUND_ONLY
    UPPER_BOUND_ONLY
    DOUBLE_BOUNDED
    FIXED

cdef extern from "<OpenMS/DATASTRUCTURES/LPWrapper.h>" namespace\
       "OpenMS::LPWrapper":
  cdef enum VariableType "OpenMS::LPWrapper::VariableType":
    #wrap-attach:
    #  LPWrapper
    CONTINUOUS
    INTEGER
    BINARY

cdef extern from "<OpenMS/DATASTRUCTURES/LPWrapper.h>" namespace\
       "OpenMS::LPWrapper":
  cdef enum Sense "OpenMS::LPWrapper::Sense":
    #wrap-attach:
    #  LPWrapper
    MIN
    MAX

cdef extern from "<OpenMS/DATASTRUCTURES/LPWrapper.h>" namespace\
       "OpenMS::LPWrapper":
  cdef enum WriteFormat "OpenMS::LPWrapper::WriteFormat":
    #wrap-attach:
```

```
    #  LPWrapper
    FORMAT_LP
    FORMAT_MPS
    FORMAT_GLPK

cdef extern from "<OpenMS/DATASTRUCTURES/LPWrapper.h>" namespace\
       "OpenMS::LPWrapper":
  cdef enum SOLVER "OpenMS::LPWrapper::SOLVER":
    #wrap-attach:
    #  LPWrapper
    SOLVER_GLPK

cdef extern from "<OpenMS/DATASTRUCTURES/LPWrapper.h>" namespace\
       "OpenMS::LPWrapper":
  cdef enum SolverStatus "OpenMS::LPWrapper::SolverStatus":
    #wrap-attach:
    #  LPWrapper
    UNDEFINED
    OPTIMAL
    FEASIBLE
    NO_FEASIBLE_SOL
```

6.191 LabeledPairFinder

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/MAPMATCHING/LabeledPairFinder.h>" namespace\
       "OpenMS":

  cdef cppclass LabeledPairFinder(BaseGroupFinder) :
    # wrap-inherits:
    #  BaseGroupFinder
    LabeledPairFinder()
    LabeledPairFinder(LabeledPairFinder)  #wrap-ignore
    void run(libcpp_vector[ ConsensusMap ] & input_maps, ConsensusMap & result_map)
    # POINTER # BaseGroupFinder * create()
    String getProductName()
```

6.192 LevMarqFitter1D

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/LevMarqFitter1D.h>"\
       namespace "OpenMS":

  cdef cppclass LevMarqFitter1D(Fitter1D):
    # wrap-ignore
    LevMarqFitter1D()
    LevMarqFitter1D(LevMarqFitter1D)
```

6.193 LightTargetedExperiment

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from\
       "<OpenMS/ANALYSIS/OPENSWATH/OPENSWATHALGO/DATAACCESS/TransitionExperiment.h>"\
       namespace "OpenSwath":

  cdef cppclass LightTransition:
    LightTransition()
    LightTransition(LightTransition)
    libcpp_string transition_name
    libcpp_string peptide_ref
    double library_intensity
    double product_mz
    double precursor_mz

    int fragment_charge
    bool decoy
    bool detecting_transition
    bool quantifying_transition
    bool identifying_transition

    int getProductChargeState()
    bool isProductChargeStateSet()
    libcpp_string getNativeID()
    libcpp_string getPeptideRef()
    double getLibraryIntensity()
    void setLibraryIntensity(double l)
    double getProductMZ()
    double getPrecursorMZ()

    void setDetectingTransition (bool d)
    bool isDetectingTransition()
    void setQuantifyingTransition (bool q)
    bool isQuantifyingTransition()
    void setIdentifyingTransition (bool i)
```

```
  bool isIdentifyingTransition()

cdef cppclass LightModification:
  LightModification()
  LightModification(LightModification)

  int location
  libcpp_string unimod_id

cdef cppclass LightCompound:
  LightCompound()
  LightCompound(LightCompound)

  double rt
  int charge
  libcpp_string sequence
  libcpp_vector[libcpp_string] protein_refs
  libcpp_string peptide_group_label
  libcpp_string id
  libcpp_string sum_formula
  libcpp_string compound_name

  libcpp_vector[LightModification] modifications

  int getChargeState()
  bool isPeptide()
  void setChargeState(int ch)

cdef cppclass LightProtein:
  LightProtein()
  LightProtein(LightProtein)
  libcpp_string id
  libcpp_string sequence

cdef cppclass LightTargetedExperiment:

  LightTargetedExperiment()
  LightTargetedExperiment(LightTargetedExperiment &)

  libcpp_vector[LightTransition] transitions
  libcpp_vector[LightCompound] compounds
  libcpp_vector[LightProtein] proteins
  libcpp_vector[LightTransition] getTransitions()

  libcpp_vector[LightCompound] getCompounds()
  libcpp_vector[LightProtein] getProteins()

  LightCompound getCompoundByRef(libcpp_string & ref)
```

```
    LightCompound getPeptideByRef(libcpp_string & ref)
```

6.194 LinearInterpolation

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/MATH/MISC/LinearInterpolation.h>" namespace\
       "OpenMS::Math":

  cdef cppclass LinearInterpolation[KeyType,ValueType]:
    # wrap-instances:
    #   LinearInterpolation := LinearInterpolation[double, double]
    LinearInterpolation()
    LinearInterpolation(LinearInterpolation)
    ValueType value(KeyType arg_pos)
    void addValue(KeyType arg_pos, ValueType arg_value)
    ValueType derivative(KeyType arg_pos)
    # TODO does this work ?
    # libcpp_vector[ValueType]  getData()
    # void setData(libcpp_vector[ValueType] & data)
    bool empty()
    KeyType key2index(KeyType pos)
    KeyType index2key(KeyType pos)
    KeyType  getScale()
    void setScale(KeyType & scale)
    KeyType  getOffset()
    void setOffset(KeyType & offset)
    void setMapping(KeyType & scale, KeyType & inside, KeyType & outside)
    void setMapping(KeyType & inside_low, KeyType & outside_low, KeyType &\
       inside_high, KeyType & outside_high)
    KeyType  getInsideReferencePoint()
    KeyType  getOutsideReferencePoint()
    KeyType supportMin()
    KeyType supportMax()
    LinearInterpolation(KeyType scale, KeyType offset)
```

6.195 LinearResampler

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/TRANSFORMERS/LinearResampler.h>" namespace\
       "OpenMS":
```

```
  cdef cppclass LinearResampler(DefaultParamHandler, ProgressLogger):
    # wrap-inherits:
    #  DefaultParamHandler
    #  ProgressLogger

    LinearResampler()
    LinearResampler(LinearResampler)    #wrap-ignore
    void raster(MSSpectrum[Peak1D] & input)
    void rasterExperiment(MSExperiment[Peak1D, ChromatogramPeak] & input)
```

6.196 LinearResamplerAlign

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/TRANSFORMERS/LinearResamplerAlign.h>" namespace\
       "OpenMS":

  cdef cppclass LinearResamplerAlign(LinearResampler) :
    # wrap-inherits:
    #  LinearResampler
    LinearResamplerAlign(LinearResamplerAlign)  #wrap-ignore
    # TEMPLATE # void raster(SpecT[ PeakType ] & spectrum)
    # TEMPLATE # void raster_align(SpecT[ PeakType ] & spectrum, double start_pos,\
       double end_pos)
    # TEMPLATE # void raster(ConstPeakTypeIterator raw_it, ConstPeakTypeIterator\
       raw_end, PeakTypeIterator resample_it, PeakTypeIterator resample_end)
    # TEMPLATE # void raster_interpolate(PeakTypeIterator raw_it, PeakTypeIterator\
       raw_end, PeakTypeIterator it, PeakTypeIterator resampled_end)
```

6.197 LocalLinearMap

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/PIP/LocalLinearMap.h>" namespace "OpenMS":

  cdef cppclass LocalLinearMap "OpenMS::LocalLinearMap":
    LocalLinearMap()
    LocalLinearMap(LocalLinearMap)  #wrap-ignore
    LLMParam getLLMParam()
    Matrix[ double ]  getCodebooks()
    Matrix[ double ]  getMatrixA()
```

```
    libcpp_vector[ double ]  getVectorWout()
    # TODO STL attributes unsigned int
    # Matrix[ UInt ]  getCord()
    void normalizeVector(libcpp_vector[ double ] & aaIndexVariables)
    # libcpp_vector[ double ] neigh(Matrix[ unsigned int ] & cord, Size win, double\
       radius)

cdef extern from "<OpenMS/ANALYSIS/PIP/LocalLinearMap.h>" namespace\
       "OpenMS::LocalLinearMap":

  cdef cppclass LLMParam "OpenMS::LocalLinearMap::LLMParam":
    LLMParam()
    LLMParam(LLMParam)  #wrap-ignore
    UInt xdim
    UInt ydim
    double radius
```

6.198 LowessSmoothing

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/SMOOTHING/LowessSmoothing.h>" namespace\
       "OpenMS":

  cdef cppclass LowessSmoothing(DefaultParamHandler):
    # wrap-inherits:
    #  DefaultParamHandler

    LowessSmoothing()
    LowessSmoothing(LowessSmoothing)

    void smoothData(libcpp_vector[double] x,
            libcpp_vector[double] y,
            libcpp_vector[double] & y_smoothed)
```

6.199 MRMAssay

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/OPENSWATH/MRMAssay.h>" namespace "OpenMS":

  cdef cppclass MRMAssay(ProgressLogger) :
```

```
    # wrap-inherits:
    #  ProgressLogger

    MRMAssay()
    MRMAssay(MRMAssay)  #wrap-ignore

    void reannotateTransitions(TargetedExperiment & exp,
                   double precursor_mz_threshold,
                   double product_mz_threshold,
                   libcpp_vector[ String ] fragment_types,
                   libcpp_vector[ size_t ] fragment_charges,
                   bool enable_reannotation,
                   bool enable_specific_losses,
                   bool enable_unspecific_losses, int round_decPow)

    void restrictTransitions(TargetedExperiment & exp,
                 double lower_mz_limit,
                 double upper_mz_limit,
                 libcpp_vector[ libcpp_pair[ double, double ] ] swathes)

    void detectingTransitions(TargetedExperiment & exp,
                  int min_transitions, int max_transitions)

    void uisTransitions(TargetedExperiment & exp,
              libcpp_vector[ String ] fragment_types,
              libcpp_vector[ size_t ] fragment_charges,
              bool enable_specific_losses,
              bool enable_unspecific_losses,
              bool enable_ms2_precursors,
              double mz_threshold,
              libcpp_vector[ libcpp_pair[ double, double ] ] swathes,
              int round_decPow,
              size_t max_num_alternative_localizations,
              int shuffle_seed)
```

6.200 MRMDecoy

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/OPENSWATH/MRMDecoy.h>" namespace "OpenMS":

  cdef cppclass MRMDecoy(ProgressLogger):
    # wrap-inherits:
    #  ProgressLogger
```

```
    MRMDecoy()
    MRMDecoy(MRMDecoy)          # wrap-ignore

    void generateDecoys(TargetedExperiment & exp,
              TargetedExperiment & dec, String method, String decoy_tag,
              double identity_threshold, int max_attempts, double mz_threshold,
              double mz_shift, bool exclude_similar,
              double similarity_threshold, bool remove_CNterm_mods,
              double precursor_mass_shift, libcpp_vector[String] fragment_types,
              libcpp_vector[size_t] fragment_charges, bool enable_specific_losses,
              bool enable_unspecific_losses, bool remove_unannotated,
              int round_decPow)
```

6.201 MRMFeature

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/KERNEL/MRMFeature.h>" namespace "OpenMS":

  cdef cppclass MRMFeature:

    MRMFeature()
    MRMFeature(MRMFeature &)

    double getScore(String name)
    void addScore(String name, double score)

    Feature getFeature(String key)
    void addFeature(Feature & f, String key)
    libcpp_vector[Feature] getFeatures()
    void getFeatureIDs(libcpp_vector[String] & result)

    Feature getPrecursorFeature(String key)
    void addPrecursorFeature(Feature & f, String key)
    void getPrecursorFeatureIDs(libcpp_vector[String] & result)
```

6.202 MRMFeatureFinderScoring

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/OPENSWATH/MRMFeatureFinderScoring.h>" namespace\
       "OpenMS":
```

```
  cdef cppclass MRMFeatureFinderScoring(DefaultParamHandler, ProgressLogger):
    # wrap-inherits:
    #  DefaultParamHandler
    #  ProgressLogger

    MRMFeatureFinderScoring()

    void pickExperiment(MSExperiment[Peak1D, ChromatogramPeak] & chromatograms,
              FeatureMap & output,
              TargetedExperiment & transition_exp_,
              TransformationDescription trafo,
              MSExperiment[Peak1D, ChromatogramPeak] & swath_map)

    void setStrictFlag(bool flag)

    void setMS1Map( shared_ptr[ SpectrumAccessOpenMS ] ms1_map)
    void prepareProteinPeptideMaps_(LightTargetedExperiment& transition_exp)
```

6.203 MRMFragmentSelection

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/MRM/MRMFragmentSelection.h>" namespace "OpenMS":

  cdef cppclass MRMFragmentSelection(DefaultParamHandler) :
    # wrap-inherits:
    #  DefaultParamHandler
    MRMFragmentSelection()
    MRMFragmentSelection(MRMFragmentSelection)
    void selectFragments(libcpp_vector[ RichPeak1D ] & selected_peaks,\
       MSSpectrum[RichPeak1D] & spec)
```

6.204 MRMIonSeries

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/OPENSWATH/MRMIonSeries.h>" namespace "OpenMS":

  cdef cppclass MRMIonSeries "OpenMS::MRMIonSeries":
    MRMIonSeries()
    MRMIonSeries(MRMIonSeries)  #wrap-ignore
```

```
    void annotateTransitionCV(ReactionMonitoringTransition & tr, String annotation)

    void annotateTransition(ReactionMonitoringTransition & tr,
                Peptide peptide, double
                precursor_mz_threshold, double
                product_mz_threshold, bool enable_reannotation,
                libcpp_vector[ String ] fragment_types,
                libcpp_vector[ size_t ] fragment_charges,
                bool enable_specific_losses,
                bool enable_unspecific_losses, int round_decPow)
```

6.205 MRMRTNormalizer

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/OPENSWATH/MRMRTNormalizer.h>" namespace\
       "OpenMS":

  cdef cppclass MRMRTNormalizer:
    pass

cdef extern from "<OpenMS/ANALYSIS/OPENSWATH/MRMRTNormalizer.h>" namespace\
       "OpenMS::MRMRTNormalizer":

  libcpp_vector[libcpp_pair[double,double]] removeOutliersIterative(
      libcpp_vector[libcpp_pair[double,double]] & pairs,
      double rsq_limit,
      double coverage_limit,
      bool use_chauvenet,
      libcpp_string outlier_detection_method
      )  # wrap-attach:MRMRTNormalizer

  libcpp_vector[libcpp_pair[double,double]] removeOutliersRANSAC(
      libcpp_vector[libcpp_pair[double,double]] & pairs,
      double rsq_limit,
      double coverage_limit,
      size_t max_iterations,
      double max_rt_threshold,
      size_t sampling_size
      )  # wrap-attach:MRMRTNormalizer

  double chauvenet_probability(libcpp_vector[ double ] residuals, int pos)  #\
       wrap-attach:MRMRTNormalizer
  bool chauvenet(libcpp_vector[ double ] residuals, int pos)  #\
```

```
       wrap-attach:MRMRTNormalizer
```

6.206 MRMScoring

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/OPENSWATH/OPENSWATHALGO/ALGO/MRMScoring.h>"\
       namespace "OpenSwath":

  cdef cppclass MRMScoring "OpenSwath::MRMScoring":
    MRMScoring(MRMScoring)  #wrap-ignore
    # TODO create class for XCorrMatrix
    # XCorrMatrixType  getXCorrMatrix()
    # NAMESPACE # # POINTER # void initializeXCorrMatrix(OpenSwath::IMRMFeature *\
       mrmfeature, OpenSwath::ITransitionGroup * transition_group, bool normalize)
    double calcXcorrCoelutionScore()
    libcpp_string calcIndXcorrIdCoelutionScore()
    double calcXcorrShape_score()
    libcpp_string calcIndXcorrIdShape_score()
    double calcXcorrShape_score_weighted(libcpp_vector[ double ] &\
       normalized_library_intensity)
    double calcXcorrCoelutionScore_weighted(libcpp_vector[ double ] &\
       normalized_library_intensity)
    # NAMESPACE # # POINTER # void calcLibraryScore(OpenSwath::IMRMFeature *\
       mrmfeature, libcpp_vector[ TransitionType ] & transitions, double & correlation,\
       double & rmsd, double & manhattan, double & dotprod)
    double calcRTScore(LightCompound & peptide, double normalized_experimental_rt)
    # NAMESPACE # # POINTER # double calcSNScore(OpenSwath::IMRMFeature *\
       mrmfeature, libcpp_vector[ OpenSwath::ISignalToNoisePtr ] &\
       signal_noise_estimators)

    double calcMS1XcorrCoelutionScore()
    double calcMS1XcorrShape_score()
```

6.207 MRMTransitionGroup

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/KERNEL/MRMTransitionGroup.h>" namespace "OpenMS":

  cdef cppclass MRMTransitionGroup[SpectrumT, TransitionT]:

    # wrap-instances:
```

```
    #   MRMTransitionGroup := MRMTransitionGroup[MSSpectrum[Peak1D],\
       ReactionMonitoringTransition]
    #   LightMRMTransitionGroup := MRMTransitionGroup[MSSpectrum[Peak1D],\
       LightTransition]

    MRMTransitionGroup()
    MRMTransitionGroup(MRMTransitionGroup[SpectrumT, TransitionT] &)

    Size size() nogil except+

    String getTransitionGroupID() nogil except+
    void setTransitionGroupID(String tr_gr_id)

    libcpp_vector[TransitionT] getTransitions() nogil except+
    libcpp_vector[TransitionT] getTransitionsMuteable()
    void addTransition(TransitionT transition, String key)
    TransitionT getTransition(String key)
    bool hasTransition(String key)

    libcpp_vector[SpectrumT] getChromatograms() nogil except+
    void addChromatogram(SpectrumT chromatogram, String key)
    SpectrumT getChromatogram(String key)
    bool hasChromatogram(String key)

    void addPrecursorChromatogram(SpectrumT chromatogram, String key)
    SpectrumT getPrecursorChromatogram(String key)
    bool hasPrecursorChromatogram(String key)

    libcpp_vector[MRMFeature] getFeatures() nogil except+
    libcpp_vector[MRMFeature] getFeaturesMuteable()
    void addFeature(MRMFeature feature)

    void getLibraryIntensity(libcpp_vector[double] result) nogil except+
    MRMTransitionGroup[SpectrumT, TransitionT] subset(libcpp_vector[ libcpp_string ]\
       tr_ids)
```

6.208 MRMTransitionGroupPicker

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/OPENSWATH/MRMTransitionGroupPicker.h>" namespace\
       "OpenMS":

  cdef cppclass MRMTransitionGroupPicker(DefaultParamHandler) :
    # wrap-inherits:
```

```
    #  DefaultParamHandler
    MRMTransitionGroupPicker()
    MRMTransitionGroupPicker(MRMTransitionGroupPicker)  #wrap-ignore
    # TEMPLATE # void pickTransitionGroup(MRMTransitionGroup[ SpectrumT, TransitionT\
       ] & transition_group)
    # TEMPLATE # MRMFeature createMRMFeature(MRMTransitionGroup[ SpectrumT,\
       TransitionT ] & transition_group, libcpp_vector[ SpectrumT ] & picked_chroms, int &\
       chr_idx, int & peak_idx)
    # TEMPLATE # void remove_overlapping_features(libcpp_vector[ SpectrumT ] &\
       picked_chroms, double best_left, double best_right)
    void findLargestPeak(libcpp_vector[ MSSpectrum[ChromatogramPeak] ] &\
       picked_chroms, int & chr_idx, int & peak_idx)
```

6.209 MS2File

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/MS2File.h>" namespace "OpenMS":

  cdef cppclass MS2File(ProgressLogger) :
    # wrap-inherits:
    #  ProgressLogger
    MS2File()
    MS2File(MS2File)  #wrap-ignore
    void load(String & filename, MSExperiment[Peak1D, ChromatogramPeak] & exp)
```

6.210 MSChromatogram

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/KERNEL/MSChromatogram.h>" namespace "OpenMS":

  cdef cppclass MSChromatogram[ChromatogramPeakT] (ChromatogramSettings,\
       MetaInfoInterface, RangeManager1):
    # wrap-inherits:
    #  ChromatogramSettings
    #  MetaInfoInterface
    #  RangeManager1

    # wrap-instances:
    #   MSChromatogram := MSChromatogram[ChromatogramPeak]
```

```
    MSChromatogram()
    MSChromatogram(MSChromatogram &)
    double getMZ()
    libcpp_string getName()
    void setName(libcpp_string)

    Size size()
    ChromatogramPeakT operator[](int)

    void updateRanges()
    void clear(int)
    void push_back(ChromatogramPeakT)

    bool isSorted()

    void sortByIntensity(bool reverse)
    void sortByPosition()

    int findNearest(double) nogil except+

    void assign(libcpp_vector[ChromatogramPeak].iterator,\
       libcpp_vector[ChromatogramPeak].iterator)  # wrap-ignore
    libcpp_vector[ChromatogramPeakT].iterator begin()   #\
       wrap-iter-begin:__iter__(ChromatogramPeakT)
    libcpp_vector[ChromatogramPeakT].iterator end()     #\
       wrap-iter-end:__iter__(ChromatogramPeakT)

    void getKeys(libcpp_vector[String] & keys)
    void getKeys(libcpp_vector[unsigned int] & keys)  # wrap-as:getKeysAsIntegers
    DataValue getMetaValue(unsigned int)
    DataValue getMetaValue(String)
    void setMetaValue(unsigned int, DataValue)
    void setMetaValue(String, DataValue)
    bool metaValueExists(String)
    bool metaValueExists(unsigned int)
    void removeMetaValue(String)
    void removeMetaValue(unsigned int)
```

6.211 MSDataCachedConsumer

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/DATAACCESS/MSDataCachedConsumer.h>" namespace\
       "OpenMS":
```

```
  cdef cppclass MSDataCachedConsumer:

    MSDataCachedConsumer(String filename, bool clear)
    MSDataCachedConsumer(MSDataCachedConsumer)  #wrap-ignore

    void consumeSpectrum(MSSpectrum[Peak1D] & s)
    void consumeChromatogram(MSChromatogram[ChromatogramPeak] & c)

    void setExperimentalSettings(ExperimentalSettings& exp)
    void setExpectedSize(Size expectedSpectra, Size expectedChromatograms)
```

6.212 MSDataWritingConsumer

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/DATAACCESS/MSDataWritingConsumer.h>" namespace\
       "OpenMS":

  cdef cppclass PlainMSDataWritingConsumer:

    PlainMSDataWritingConsumer(String filename)
    PlainMSDataWritingConsumer(PlainMSDataWritingConsumer)  #wrap-ignore

    void consumeSpectrum(MSSpectrum[Peak1D] & s)
    void consumeChromatogram(MSChromatogram[ChromatogramPeak] & c)

    void setExperimentalSettings(ExperimentalSettings& exp)
    void setExpectedSize(Size expectedSpectra, Size expectedChromatograms)

    void addDataProcessing(DataProcessing d)
    Size getNrSpectraWritten()
    Size getNrChromatogramsWritten()

  cdef cppclass NoopMSDataWritingConsumer:

    NoopMSDataWritingConsumer(String filename)
    NoopMSDataWritingConsumer(NoopMSDataWritingConsumer)  #wrap-ignore

    void consumeSpectrum(MSSpectrum[Peak1D] & s)
    void consumeChromatogram(MSChromatogram[ChromatogramPeak] & c)

    void setExperimentalSettings(ExperimentalSettings& exp)
    void setExpectedSize(Size expectedSpectra, Size expectedChromatograms)

    void addDataProcessing(DataProcessing d)
```

```
    Size getNrSpectraWritten()
    Size getNrChromatogramsWritten()
```

6.213 MSExperiment

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/KERNEL/MSExperiment.h>" namespace "OpenMS":

  cdef cppclass MSExperiment[PeakT, ChromoPeakT](ExperimentalSettings,\
      RangeManager2):
    # wrap-inherits:
    #   ExperimentalSettings
    #   RangeManager2
    #
    # wrap-instances:
    #   MSExperiment := MSExperiment[Peak1D, ChromatogramPeak]
    #   RichMSExperiment := MSExperiment[RichPeak1D, ChromatogramPeak]

    MSExperiment()
    MSExperiment(MSExperiment[PeakT, ChromoPeakT] &)

    bool operator==(MSExperiment[PeakT, ChromatogramPeak])
    void reset()
    bool clearMetaDataArrays()
    ExperimentalSettings getExperimentalSettings()

    StringList getPrimaryMSRunPath()

    void swap(MSExperiment[PeakT, ChromoPeakT])

    void addSpectrum(MSSpectrum[PeakT] spec)
    MSSpectrum[PeakT] operator[](int)     # wrap-upper-limit:size()  # TODO\
       deprecate for 1.12
    MSSpectrum[PeakT] getSpectrum(Size id_)
    void setSpectra(libcpp_vector[ MSSpectrum[ PeakT ] ] & spectra)
    libcpp_vector[MSSpectrum[PeakT]] getSpectra()   # TODO deprecate for 1.12

    libcpp_vector[MSSpectrum[PeakT]].iterator begin()     #\
       wrap-iter-begin:__iter__(MSSpectrum[PeakT])
    libcpp_vector[MSSpectrum[PeakT]].iterator end()       #\
       wrap-iter-end:__iter__(MSSpectrum[PeakT])

    MSChromatogram[ ChromoPeakT ]  getChromatogram(Size id_)
    void addChromatogram(MSChromatogram[ChromoPeakT] chromatogram)
```

```
void setChromatograms(libcpp_vector[MSChromatogram[ChromoPeakT]] chromatograms)
libcpp_vector[MSChromatogram[ChromoPeakT]] getChromatograms()  # TODO deprecate\
   for 1.12

MSChromatogram[ChromoPeakT] getTIC()
void clear(bool clear_meta_data)

void updateRanges()
void updateRanges(int msLevel)

void reserveSpaceSpectra(Size s)
void reserveSpaceChromatograms(Size s)

double getMinMZ()
double getMaxMZ()
double getMinRT()
double getMaxRT()

UInt64 getSize()
int   size()  # TODO deprecate for 1.12
void resize(Size s)
bool empty()
void reserve(Size s)
Size getNrSpectra()
Size getNrChromatograms()
libcpp_vector[unsigned int] getMSLevels()   # wrap-ignore

void sortSpectra(bool sort_mz)
void sortSpectra()
void sortChromatograms(bool sort_rt)
void sortChromatograms()

bool isSorted(bool check_mz)
bool isSorted()

void getKeys(libcpp_vector[String] & keys)
void getKeys(libcpp_vector[unsigned int] & keys)  # wrap-as:getKeysAsIntegers
DataValue getMetaValue(unsigned int)
DataValue getMetaValue(String)
void setMetaValue(unsigned int, DataValue)
void setMetaValue(String, DataValue)
bool metaValueExists(String)
bool metaValueExists(unsigned int)
void removeMetaValue(String)
void removeMetaValue(unsigned int)
```

6.214 MSNumpressCoder

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/MSNumpressCoder.h>" namespace "OpenMS":

  cdef cppclass MSNumpressCoder:

    MSNumpressCoder()
    MSNumpressCoder(MSNumpressCoder)

    void encodeNP(libcpp_vector[double] in_, String & result,
        bool zlib_compression, NumpressConfig config)

    void decodeNP(String in_, libcpp_vector[double] & out,
        bool zlib_compression, NumpressConfig config)

cdef extern from "<OpenMS/FORMAT/MSNumpressCoder.h>" namespace\
       "OpenMS::MSNumpressCoder":

  cdef enum NumpressCompression:
    # wrap-attach:
    #   MSNumpressCoder
    NONE,
    LINEAR,
    PIC,
    SLOF,
    SIZE_OF_NUMPRESSCOMPRESSION

  cdef cppclass NumpressConfig:
    # wrap-attach:
    #   MSNumpressCoder

    NumpressConfig()
    NumpressConfig(NumpressConfig)

    double numpressFixedPoint
    double numpressErrorTolerance
    NumpressCompression np_compression
    bool estimate_fixed_point

    void setCompression(libcpp_string & compression)
```

6.215 MSPFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/MSPFile.h>" namespace "OpenMS":

  cdef cppclass MSPFile:

    MSPFile()

    void store(String filename, MSExperiment[RichPeak1D, ChromatogramPeak] & exp)
    void load(String filename, libcpp_vector[PeptideIdentification] & ids,\
       MSExperiment[RichPeak1D, ChromatogramPeak] & exp)
```

6.216 MSQuantifications

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/MSQuantifications.h>" namespace "OpenMS":

  cdef cppclass MSQuantifications(ExperimentalSettings):
    # wrap-inherits:
    #  ExperimentalSettings
    MSQuantifications()  nogil
    MSQuantifications(MSQuantifications) nogil

    bool operator==(MSQuantifications &) nogil
    bool operator!=(MSQuantifications &) nogil

    libcpp_vector[DataProcessing] getDataProcessingList()
    libcpp_vector[Assay] getAssays()
    # libcpp_map[String, Ratio] getRatios()  # wrap-ignore
    libcpp_vector[ConsensusMap] getConsensusMaps()
    void setConsensusMaps(libcpp_vector[ConsensusMap])
    libcpp_vector[FeatureMap ] getFeatureMaps()
    AnalysisSummary getAnalysisSummary()
    void setDataProcessingList(libcpp_vector[DataProcessing] dpl)
    void setAnalysisSummaryQuantType(QUANT_TYPES r)
    void addConsensusMap(ConsensusMap m)
    void assignUIDs()
    void registerExperiment(MSExperiment[Peak1D, ChromatogramPeak] exp,
                libcpp_vector[ libcpp_vector[ libcpp_pair[
                  String, double] ] ] labels)  # wrap-ignore

cdef extern from "<OpenMS/METADATA/MSQuantifications.h>" namespace\
```

```
      "OpenMS::MSQuantifications":
  cdef enum QUANT_TYPES:
    # wrap-attach:
    #   MSQuantifications
    MS1LABEL = 0,
    MS2LABEL,
    LABELFREE,
    SIZE_OF_QUANT_TYPES

  cdef cppclass AnalysisSummary:
    # wrap-attach:
    #   MSQuantifications
    AnalysisSummary()
    AnalysisSummary(AnalysisSummary)

    MetaInfo user_params_
    CVTermList cv_params_
    QUANT_TYPES quant_type_

  cdef cppclass Assay:
    # wrap-attach:
    #   MSQuantifications
    Assay()
    Assay(Assay)

    String uid_
    libcpp_vector[libcpp_pair[String, double] ] mods_ # wrap-ignore
    libcpp_vector[ExperimentalSettings] raw_files_
    libcpp_map[size_t, FeatureMap ] feature_maps_
    # iTRAQ needs no FeatureMaps so ExperimentalSettings are not directly mapped to\
      FeatureMaps
```

6.217 MSSim

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/SIMULATION/MSSim.h>" namespace "OpenMS":

  cdef cppclass MSSim:

    MSSim()
    MSSim(MSSim)  # wrap-ignore

    void simulate(shared_ptr[SimRandomNumberGenerator] rnd_gen, SampleChannels\
      peptides)
```

```
    MSExperiment[Peak1D, ChromatogramPeak] getExperiment()

    FeatureMap getSimulatedFeatures()

    ConsensusMap getChargeConsensus()

    FeatureMap getContaminants()

    ConsensusMap getLabelingConsensus()

    MSExperiment[Peak1D, ChromatogramPeak] getPeakMap()

    Param getParameters()

    void getMS2Identifications(libcpp_vector[ ProteinIdentification ] & proteins,\
       libcpp_vector[ PeptideIdentification ] & peptides)
```

6.218 MSSpectrum

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/KERNEL/MSSpectrum.h>" namespace "OpenMS":

  cdef cppclass MSSpectrum[PeakT](SpectrumSettings, MetaInfoInterface,\
      RangeManager1):
    # wrap-inherits:
    #  SpectrumSettings
    #  MetaInfoInterface
    #  RangeManager1

    # wrap-instances:
    #   MSSpectrum := MSSpectrum[Peak1D]
    #   RichMSSpectrum := MSSpectrum[RichPeak1D]
    #   ChromatogramSpectrum := MSSpectrum[ChromatogramPeak]

    MSSpectrum()
    MSSpectrum(MSSpectrum[PeakT] &)
    double getRT()
    void   setRT(double)
    unsigned int getMSLevel()
    void setMSLevel(unsigned int)

    libcpp_string getName()
    void setName(libcpp_string)
```

```
    Size size()

    PeakT operator[](int)  # wrap-upper-limit:size()

    void updateRanges()
    void clear(int)
    void push_back(PeakT)

    bool isSorted()

    int findNearest(double) nogil except+
    int findNearest(double, double) nogil except+
    int findNearest(double, double, double) nogil except+

    void assign(libcpp_vector[Peak1D].iterator, libcpp_vector[Peak1D].iterator)  #\
       wrap-ignore
    libcpp_vector[PeakT].iterator begin()   # wrap-iter-begin:__iter__(PeakT)
    libcpp_vector[PeakT].iterator end()     # wrap-iter-end:__iter__(PeakT)

    bool operator==(MSSpectrum[PeakT])
    bool operator!=(MSSpectrum[PeakT])

    void sortByIntensity(bool reverse)
    void sortByPosition()

    void getKeys(libcpp_vector[String] & keys)
    void getKeys(libcpp_vector[unsigned int] & keys)  # wrap-as:getKeysAsIntegers
    DataValue getMetaValue(unsigned int)
    DataValue getMetaValue(String)
    void setMetaValue(unsigned int, DataValue)
    void setMetaValue(String, DataValue)
    bool metaValueExists(String)
    bool metaValueExists(unsigned int)
    void removeMetaValue(String)
    void removeMetaValue(unsigned int)
```

6.219 MZTrafoModel

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/CALIBRATION/MZTrafoModel.h>" namespace\
       "OpenMS":

  cdef cppclass MZTrafoModel:
    MZTrafoModel()
```

```
    MZTrafoModel(MZTrafoModel &)
    MZTrafoModel(bool)
    bool isTrained()
    double getRT()
    double predict(double)
    bool train(CalibrationData, MZTrafoModel_MODELTYPE, bool, double, double)
    bool train(libcpp_vector[double], libcpp_vector[double], libcpp_vector[double],\
       MZTrafoModel_MODELTYPE, bool)
    void getCoefficients(double& intercept, double& slope, double& power)
    void setCoefficients(MZTrafoModel)
    void setCoefficients(double, double, double)
    String toString()

cdef extern from "<OpenMS/FILTERING/CALIBRATION/MZTrafoModel.h>" namespace\
       "OpenMS::MZTrafoModel":

  cdef enum MZTrafoModel_MODELTYPE "OpenMS::MZTrafoModel::MODELTYPE":
    LINEAR
    LINEAR_WEIGHTED
    QUADRATIC
    QUADRATIC_WEIGHTED
    SIZE_OF_MODELTYPE

  MZTrafoModel_MODELTYPE nameToEnum(libcpp_string name)  # wrap-attach:MZTrafoModel
  libcpp_string enumToName(MZTrafoModel_MODELTYPE mt)  # wrap-attach:MZTrafoModel
  void setRANSACParams(RANSACParam p)  # wrap-attach:MZTrafoModel
  void setCoefficientLimits(double offset, double scale, double power)  #\
       wrap-attach:MZTrafoModel
  bool isValidModel(MZTrafoModel& trafo)  # wrap-attach:MZTrafoModel
  Size findNearest(libcpp_vector[MZTrafoModel]& tms, double rt)  #\
       wrap-attach:MZTrafoModel
```

6.220 Map

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/DATASTRUCTURES/Map.h>" namespace "OpenMS":

  cdef cppclass Map[U, V]:
    # wrap-ignore

    U first
    V second

    Map()
```

```
    Map(Map[U,V])

    V& operator[](U&)  nogil

    cppclass iterator:
      bool operator!=(Map.iterator) nogil
      Map operator*() nogil
      Map operator++() nogil

    Map.iterator begin() nogil
    Map.iterator end() nogil
```

6.221 MapAlignmentAlgorithmIdentification

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from\
       "<OpenMS/ANALYSIS/MAPMATCHING/MapAlignmentAlgorithmIdentification.h>" namespace\
       "OpenMS":

  cdef cppclass MapAlignmentAlgorithmIdentification(DefaultParamHandler,\
       ProgressLogger):
    # wrap-inherits:
    #  DefaultParamHandler
    #  ProgressLogger

    MapAlignmentAlgorithmIdentification()

    void align(libcpp_vector[MSExperiment[Peak1D,ChromatogramPeak]]&,\
       libcpp_vector[TransformationDescription]&, int)
    void align(libcpp_vector[FeatureMap]&,\
       libcpp_vector[TransformationDescription]&, int)
    void align(libcpp_vector[ConsensusMap]&,\
       libcpp_vector[TransformationDescription]&, int)
    void align(libcpp_vector[libcpp_vector[PeptideIdentification]]& ids,\
       libcpp_vector[TransformationDescription]& trafos, int ref_index)  #wrap-ignore

    void setReference(MSExperiment[Peak1D,ChromatogramPeak]&)
    void setReference(FeatureMap&)
    void setReference(ConsensusMap&)
    void setReference(libcpp_vector[PeptideIdentification]&)
```

6.222 MapAlignmentAlgorithmPoseClustering

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from\
       "<OpenMS/ANALYSIS/MAPMATCHING/MapAlignmentAlgorithmPoseClustering.h>" namespace\
       "OpenMS":

  cdef cppclass MapAlignmentAlgorithmPoseClustering(DefaultParamHandler,\
       ProgressLogger):
    # wrap-inherits:
    #  DefaultParamHandler
    #  ProgressLogger

    MapAlignmentAlgorithmPoseClustering()

    void align(FeatureMap,
           TransformationDescription &
           )

    void align(MSExperiment[Peak1D,ChromatogramPeak],
           TransformationDescription &
           )

    void setReference (FeatureMap)
    void setReference (MSExperiment[Peak1D,ChromatogramPeak])
```

6.223 MapAlignmentAlgorithmSpectrumAlignment

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from\
       "<OpenMS/ANALYSIS/MAPMATCHING/MapAlignmentAlgorithmSpectrumAlignment.h>" namespace\
       "OpenMS":

  cdef cppclass MapAlignmentAlgorithmSpectrumAlignment(DefaultParamHandler,\
       ProgressLogger):
    # wrap-inherits:
    #  DefaultParamHandler
    #  ProgressLogger

    MapAlignmentAlgorithmSpectrumAlignment()

    void align(libcpp_vector[MSExperiment[Peak1D, ChromatogramPeak]]&,\
```

```
       libcpp_vector[TransformationDescription]&)
```

6.224 MapAlignmentEvaluationAlgorithm

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/MAPMATCHING/MapAlignmentEvaluationAlgorithm.h>"\
       namespace "OpenMS":

  cdef cppclass MapAlignmentEvaluationAlgorithm\
       "OpenMS::MapAlignmentEvaluationAlgorithm":
    # wrap-ignore
    # ABSTRACT class
    MapAlignmentEvaluationAlgorithm()
    MapAlignmentEvaluationAlgorithm(MapAlignmentEvaluationAlgorithm)  #wrap-ignore
    # NAMESPACE # void evaluate(ConsensusMap & conensus_map_in, ConsensusMap &\
       consensus_map_gt, double & rt_dev, double & mz_dev, Peak2D::IntensityType &\
       int_dev, bool use_charge, double & out)
    # NAMESPACE # bool isSameHandle(FeatureHandle & lhs, FeatureHandle & rhs, double\
       & rt_dev, double & mz_dev, Peak2D::IntensityType & int_dev, bool use_charge)
    void registerChildren()
```

6.225 MapAlignmentEvaluationAlgorithmPrecision

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from\
       "<OpenMS/ANALYSIS/MAPMATCHING/MapAlignmentEvaluationAlgorithmPrecision.h>"\
       namespace "OpenMS":

  cdef cppclass\
       MapAlignmentEvaluationAlgorithmPrecision(MapAlignmentEvaluationAlgorithm) :
    # wrap-inherits:
    #  MapAlignmentEvaluationAlgorithm
    MapAlignmentEvaluationAlgorithmPrecision()
   \
       MapAlignmentEvaluationAlgorithmPrecision(MapAlignmentEvaluationAlgorithmPrecision) \
       #wrap-ignore
    # NAMESPACE # void evaluate(ConsensusMap & consensus_map_in, ConsensusMap &\
       consensus_map_gt, double & rt_dev, double & mz_dev, Peak2D::IntensityType &\
       int_dev, bool use_charge, double & out)
    # POINTER # MapAlignmentEvaluationAlgorithm * create()
```

```
    String getProductName()
```

6.226 MapAlignmentEvaluationAlgorithmRecall

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from\
       "<OpenMS/ANALYSIS/MAPMATCHING/MapAlignmentEvaluationAlgorithmRecall.h>" namespace\
       "OpenMS":

  cdef cppclass\
       MapAlignmentEvaluationAlgorithmRecall(MapAlignmentEvaluationAlgorithm) :
    # wrap-inherits:
    #  MapAlignmentEvaluationAlgorithm
    MapAlignmentEvaluationAlgorithmRecall()
    MapAlignmentEvaluationAlgorithmRecall(MapAlignmentEvaluationAlgorithmRecall) \
       #wrap-ignore
    # NAMESPACE # void evaluate(ConsensusMap & consensus_map_in, ConsensusMap &\
       consensus_map_gt, double & rt_dev, double & mz_dev, Peak2D::IntensityType &\
       int_dev, bool use_charge, double & out)
    # POINTER # MapAlignmentEvaluationAlgorithm * create()
    String getProductName()
```

6.227 MapAlignmentTransformer

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/MAPMATCHING/MapAlignmentTransformer.h>" namespace\
       "OpenMS":

  cdef cppclass MapAlignmentTransformer:

    MapAlignmentTransformer()

    void transformRetentionTimes(MSExperiment[Peak1D,ChromatogramPeak]&,\
       TransformationDescription&, bool)

    void transformRetentionTimes(FeatureMap&, TransformationDescription&, bool)

    void transformRetentionTimes(ConsensusMap&, TransformationDescription&, bool)

    void transformRetentionTimes(libcpp_vector[PeptideIdentification]&,\
```

```
       TransformationDescription&, bool)
```

6.228 MarkerMower

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/TRANSFORMERS/MarkerMower.h>" namespace\
       "OpenMS":

  cdef cppclass MarkerMower(DefaultParamHandler):
    # wrap-inherits:
    #  DefaultParamHandler

    MarkerMower()
    MarkerMower(MarkerMower)  #wrap-ignore

    void filterSpectrum(MSSpectrum[Peak1D] & spec)
    void filterPeakSpectrum(MSSpectrum[Peak1D] & spec)
    void filterPeakMap(MSExperiment[Peak1D, ChromatogramPeak] & exp)

    String getProductName()

    void insertmarker(PeakMarker * peak_marker)
```

6.229 MascotGenericFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/MascotGenericFile.h>" namespace "OpenMS":

  cdef cppclass MascotGenericFile(ProgressLogger, DefaultParamHandler) :
    # wrap-inherits:
    #  ProgressLogger
    #  DefaultParamHandler
    MascotGenericFile()
    MascotGenericFile(MascotGenericFile)  #wrap-ignore
    void store(String & filename, MSExperiment[Peak1D, ChromatogramPeak] &\
       experiment)
    # NAMESPACE # void store(std::ostream & os, String & filename,\
       MSExperiment[Peak1D, ChromatogramPeak] & experiment)
    void load(String & filename, MSExperiment[Peak1D, ChromatogramPeak] & exp)
    libcpp_pair[ String, String ] getHTTPPeakListEnclosure(String & filename)
```

```
    void updateMembers_()
```

6.230 MascotInfile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/MascotInfile.h>" namespace "OpenMS":

  cdef cppclass MascotInfile(ProgressLogger) :
    # wrap-inherits:
    #  ProgressLogger
    MascotInfile()
    MascotInfile(MascotInfile)  #wrap-ignore
    void store(String & filename, MSSpectrum[Peak1D] & spec, double mz, double\
       retention_time, String search_title)
    void store(String & filename, MSExperiment[Peak1D, ChromatogramPeak] &\
       experiment, String search_title)
    void load(String & filename, MSExperiment[Peak1D, ChromatogramPeak] & exp)
    String  getBoundary()
    void setBoundary(String & boundary)
    String  getDB()
    void setDB(String & db)
    String  getSearchType()
    void setSearchType(String & search_type)
    String  getHits()
    void setHits(String & hits)
    String  getCleavage()
    void setCleavage(String & cleavage)
    String  getMassType()
    void setMassType(String & mass_type)
    libcpp_vector[ String ]  getModifications()
    void setModifications(libcpp_vector[ String ] & mods)
    libcpp_vector[ String ]  getVariableModifications()
    void setVariableModifications(libcpp_vector[ String ] & mods)
    String  getInstrument()
    void setInstrument(String & instrument)
    UInt getMissedCleavages()
    void setMissedCleavages(UInt missed_cleavages)
    float getPrecursorMassTolerance()
    void setPrecursorMassTolerance(float precursor_mass_tolerance)
    float getPeakMassTolerance()
    void setPeakMassTolerance(float ion_mass_tolerance)
    String  getTaxonomy()
    void setTaxonomy(String & taxonomy)
    String  getFormVersion()
```

```
    void setFormVersion(String & form_version)
    String  getCharges()
    void setCharges(libcpp_vector[ int ] & charges)
```

6.231 MascotXMLFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/MascotXMLFile.h>" namespace "OpenMS":

  cdef cppclass MascotXMLFile(XMLFile) :
    # wrap-inherits:
    #  XMLFile
    MascotXMLFile()
    MascotXMLFile(MascotXMLFile)  #wrap-ignore
```

6.232 MassAnalyzer

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/MassAnalyzer.h>" namespace "OpenMS":

  cdef cppclass MassAnalyzer(MetaInfoInterface):
    # wrap-inherits:
    #  MetaInfoInterface

    MassAnalyzer()
    MassAnalyzer(MassAnalyzer)  # wrap-ignore

    AnalyzerType getType()
    void setType(AnalyzerType type)

    ResolutionMethod getResolutionMethod()
    void setResolutionMethod(ResolutionMethod resolution_method)

    ResolutionType getResolutionType()
    void setResolutionType(ResolutionType resolution_type)

    ScanDirection getScanDirection()
    void setScanDirection(ScanDirection scan_direction)

    ScanLaw getScanLaw()
```

```
    void setScanLaw(ScanLaw scan_law)

    ReflectronState getReflectronState()
    void setReflectronState(ReflectronState reflecton_state)

    double getResolution()
    void setResolution(double resolution)

    double getAccuracy()
    void setAccuracy(double accuracy)

    double getScanRate()
    void setScanRate(double scan_rate)

    double getScanTime()
    void setScanTime(double scan_time)

    double getTOFTotalPathLength()
    void setTOFTotalPathLength(double TOF_total_path_length)

    double getIsolationWidth()
    void setIsolationWidth(double isolation_width)

    Int getFinalMSExponent()
    void setFinalMSExponent(Int final_MS_exponent)

    double getMagneticFieldStrength()
    void setMagneticFieldStrength(double magnetic_field_strength)

    Int getOrder()
    void setOrder(Int order)

cdef extern from "<OpenMS/METADATA/MassAnalyzer.h>" namespace\
       "OpenMS::MassAnalyzer":

  cdef enum AnalyzerType:
    # wrap-attach:
    #   MassAnalyzer
    ANALYZERNULL,                    #< Unknown
    QUADRUPOLE,                      #< Quadrupole
    PAULIONTRAP,                     #< Quadrupole ion trap / Paul ion trap
    RADIALEJECTIONLINEARIONTRAP,     #< Radial ejection linear ion trap
    AXIALEJECTIONLINEARIONTRAP,      #< Axial ejection linear ion trap
    TOF,                             #< Time-of-flight
    SECTOR,                          #< Magnetic sector
    FOURIERTRANSFORM,                #< Fourier transform ion cyclotron resonance\
       mass spectrometer
    IONSTORAGE,                      #< Ion storage
```

```
  ESA,                              #< Electrostatic energy analyzer
  IT,                               #< Ion trap
  SWIFT,                           #< Stored waveform inverse fourier transform
  CYCLOTRON,                       #< Cyclotron
  ORBITRAP,                         #< Orbitrap
  LIT,                              #< Linear ion trap
  SIZE_OF_ANALYZERTYPE

cdef enum ResolutionMethod:
  # wrap-attach:
  #   MassAnalyzer
  RESMETHNULL,            #< Unknown
  FWHM,                   #< Full width at half max
  TENPERCENTVALLEY,       #< Ten percent valley
  BASELINE,               #< Baseline
  SIZE_OF_RESOLUTIONMETHOD

cdef enum ResolutionType:
  # wrap-attach:
  #   MassAnalyzer
  RESTYPENULL,        #< Unknown
  CONSTANT,           #< Constant
  PROPORTIONAL,       #< Proportional
  SIZE_OF_RESOLUTIONTYPE

cdef enum ScanDirection:
  # wrap-attach:
  #   MassAnalyzer
  SCANDIRNULL,        #< Unknown
  UP,                 #< Up
  DOWN,               #< Down
  SIZE_OF_SCANDIRECTION

cdef enum ScanLaw:
  # wrap-attach:
  #   MassAnalyzer
  SCANLAWNULL,        #< Unknown
  EXPONENTIAL,        #< Unknown
  LINEAR,             #< Linear
  QUADRATIC,        #< Quadratic
  SIZE_OF_SCANLAW

cdef enum ReflectronState:
  # wrap-attach:
  #   MassAnalyzer
  REFLSTATENULL,      #< Unknown
  ON,                   #< On
  OFF,                  #< Off
```

```
    NONE,                 #< None
    SIZE_OF_REFLECTRONSTATE
```

6.233 MassDecomposer

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/MASSDECOMPOSITION/IMS/MassDecomposer.h>"\
       namespace "OpenMS::ims":

  cdef cppclass MassDecomposer[ValueType,DecompositionValueType]:
    # wrap-ignore
    # ABSTRACT class
    MassDecomposer(MassDecomposer)  #wrap-ignore
    bool exist(ValueType mass)
    # decomposition_type getDecomposition(ValueType mass)
    # decompositions_type getAllDecompositions(ValueType mass)
    DecompositionValueType getNumberOfDecompositions(ValueType mass)
```

6.234 MassDecomposition

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/MASSDECOMPOSITION/MassDecomposition.h>"\
       namespace "OpenMS":

  cdef cppclass MassDecomposition "OpenMS::MassDecomposition":
    MassDecomposition()
    MassDecomposition(MassDecomposition)
    MassDecomposition(String & deco)
    String toString()
    String toExpandedString()
    Size getNumberOfMaxAA()
    bool containsTag(String & tag)
    bool compatible(MassDecomposition & deco)
```

6.235 MassDecompositionAlgorithm

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/MASSDECOMPOSITION/MassDecompositionAlgorithm.h>"\
       namespace "OpenMS":

  cdef cppclass MassDecompositionAlgorithm(DefaultParamHandler) :
    # wrap-inherits:
    #  DefaultParamHandler
    MassDecompositionAlgorithm()
    MassDecompositionAlgorithm(MassDecompositionAlgorithm)  #wrap-ignore
    void getDecompositions(libcpp_vector[ MassDecomposition ] & decomps, double\
       weight)
```

6.236 MassExplainer

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
ctypedef libcpp_vector[Adduct] AdductsType

cdef extern from "<OpenMS/DATASTRUCTURES/MassExplainer.h>" namespace "OpenMS":

  cdef cppclass MassExplainer "OpenMS::MassExplainer":
    MassExplainer()
    MassExplainer(MassExplainer)  #wrap-ignore
    MassExplainer(libcpp_vector[Adduct] adduct_base)
    MassExplainer(Int q_min, Int q_max, Int max_span, double thresh_logp)
    ## MassExplainer(libcpp_vector[Adduct] adduct_base, Int q_min, Int q_max, Int\
       max_span, double thresh_logp, Size max_neutrals)
    ## MassExplainer operator=(MassExplainer &rhs)
    void setAdductBase(libcpp_vector[Adduct] adduct_base)
    libcpp_vector[Adduct] getAdductBase()
    Compomer getCompomerById(Size id)
    void compute()
```

6.237 MassTrace

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/KERNEL/MassTrace.h>" namespace "OpenMS":

  cdef cppclass Kernel_MassTrace "OpenMS::MassTrace":

    Kernel_MassTrace()
    Kernel_MassTrace(Kernel_MassTrace &)
```

```
    Size getSize()
    String getLabel()
    void setLabel(String label)

    double getCentroidMZ()
    double getCentroidRT()
    double getCentroidSD()
    double getFWHM()
    double getTraceLength()
    libcpp_pair[Size,Size] getFWHMborders()
    libcpp_vector[double] getSmoothedIntensities()
    double getAverageMS1CycleTime()

    double computeSmoothedPeakArea()
    double computePeakArea()
    Size findMaxByIntPeak(bool)
    Size estimateFWHM(bool)
    double computeFwhmArea()
    double computeFwhmAreaSmooth()
    double getIntensity(bool)
    double getMaxIntensity(bool)

    ConvexHull2D getConvexhull()

    void setCentroidSD(double &tmp_sd)
    void setSmoothedIntensities(libcpp_vector[ double ] &db_vec)
    void updateSmoothedMaxRT()
    void updateWeightedMeanRT()
    void updateSmoothedWeightedMeanRT()
    void updateMedianRT()
    void updateMedianMZ()
    void updateMeanMZ()
    void updateWeightedMeanMZ()
    void updateWeightedMZsd()

    void setQuantMethod(MT_QUANTMETHOD method)
    MT_QUANTMETHOD getQuantMethod()

cdef extern from "<OpenMS/KERNEL/MassTrace.h>" namespace "OpenMS::MassTrace":

  cdef enum MT_QUANTMETHOD:
    MT_QUANT_AREA,
    MT_QUANT_MEDIAN,
    SIZE_OF_MT_QUANTMETHOD
```

6.238 MassTraceDetection

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/DATAREDUCTION/MassTraceDetection.h>" namespace\
       "OpenMS":

  cdef cppclass MassTraceDetection(ProgressLogger, DefaultParamHandler):
    # wrap-inherits:
    #  ProgressLogger
    #  DefaultParamHandler

    MassTraceDetection()

    void run(MSExperiment[Peak1D, ChromatogramPeak] & input_map,
        libcpp_vector[Kernel_MassTrace] & traces
        )
```

6.239 Matrix

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/DATASTRUCTURES/Matrix.h>" namespace "OpenMS":

  cdef cppclass Matrix[ValueT]:
    # wrap-instances:
    #   MatrixDouble := Matrix[double]
    Matrix()
    Matrix(Matrix[ValueT])
    Matrix(size_t rows, size_t cols, ValueT value)
    # const_reference operator()(size_t i, size_t j)
    # reference operator()(size_t i, size_t j)
    # const_reference getValue(size_t i, size_t j)
    ValueT getValue(size_t i, size_t j)
    void setValue(size_t i, size_t j, ValueT value)
    ## The following two lines introduce an odd bug:
    # static PyObject *__pyx_convert_vector_to_py_double is declared twice by\
       Cython:
    # TODO look into Cython Bug
    # libcpp_vector[ValueT] row(size_t i)
    # libcpp_vector[ValueT] col(size_t i)
    void clear()
    void resize(size_t i, size_t j, ValueT value)
    void resize(libcpp_pair[ size_t, size_t ] & size_pair, ValueT value)
```

```
    size_t rows()
    size_t cols()
    libcpp_pair[ size_t, size_t ] sizePair()
    size_t index(size_t row, size_t col)
    libcpp_pair[ size_t, size_t ] indexPair(size_t index)
    size_t colIndex(size_t index)
    size_t rowIndex(size_t index)
    ## bool operator==(Matrix & rhs)
    ## bool operator<(Matrix & rhs)
    # TEMPLATE # void setMatrix(ValueType matrix)
    # TEMPLATE # # NAMESPACE # std::ostream  operator<[(std::ostream & os, Matrix[\
       Value ] & matrix)
    #   MatrixUnsignedInt := Matrix[unsigned int]
```

6.240 MetaInfo

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/MetaInfo.h>" namespace "OpenMS":

  cdef cppclass MetaInfo:

    MetaInfo()
    MetaInfo(MetaInfo)

    DataValue getValue(String name)
    DataValue getValue(UInt index)

    bool exists(String name)
    bool exists(UInt index)

    void setValue(String name, DataValue value)
    void setValue(UInt index, DataValue value)

    void removeValue(String name)
    void removeValue(UInt index)

    void getKeys(libcpp_vector[String] & keys)

    void getKeys(libcpp_vector[unsigned int] & keys)   # wrap-as:getKeysAsIntegers

    bool empty()

    void clear()
```

```
    MetaInfoRegistry  registry()
```

6.241 MetaInfoInterface

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/MetaInfoInterface.h>" namespace "OpenMS":

  cdef cppclass MetaInfoInterface:

    MetaInfoInterface()
    MetaInfoInterface(MetaInfoInterface)

    bool operator==(MetaInfoInterface)
    bool operator!=(MetaInfoInterface)
    bool isMetaEmpty()
    void clearMetaInfo()

    MetaInfoRegistry metaRegistry()
```

6.242 MetaInfoRegistry

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/MetaInfoRegistry.h>" namespace "OpenMS":

  cdef cppclass MetaInfoRegistry "OpenMS::MetaInfoRegistry":
    MetaInfoRegistry()
    MetaInfoRegistry(MetaInfoRegistry)
    UInt registerName(String & name, String & description, String & unit)
    void setDescription(UInt index, String & description)
    void setDescription(String & name, String & description)
    void setUnit(UInt index, String & unit)
    void setUnit(String & name, String & unit)
    UInt getIndex(String & name)
    String getName(UInt index)
    String getDescription(UInt index)
    String getDescription(String & name)
    String getUnit(UInt index)
    String getUnit(String & name)
```

6.243 MetaboliteSpectralMatching

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/ID/MetaboliteSpectralMatching.h>" namespace\
       "OpenMS":

  cdef cppclass MetaboliteSpectralMatching(ProgressLogger, DefaultParamHandler):
    # wrap-inherits:
    #  ProgressLogger
    #  DefaultParamHandler

    MetaboliteSpectralMatching()
    MetaboliteSpectralMatching(MetaboliteSpectralMatching)

    double computeHyperScore(MSSpectrum[Peak1D], MSSpectrum[Peak1D], double, double)

    void run(MSExperiment[Peak1D,ChromatogramPeak] & exp, MzTab& mz_tab)

  cdef cppclass SpectralMatch:

    SpectralMatch()
    SpectralMatch(SpectralMatch)

    double getObservedPrecursorMass()
    void setObservedPrecursorMass(double)

    double getObservedPrecursorRT()
    void setObservedPrecursorRT(double)

    double getFoundPrecursorMass()
    void setFoundPrecursorMass(double)

    Int getFoundPrecursorCharge()
    void setFoundPrecursorCharge(Int)

    double getMatchingScore()
    void setMatchingScore(double)

    Size getObservedSpectrumIndex()
    void setObservedSpectrumIndex(Size)

    Size getMatchingSpectrumIndex()
    void setMatchingSpectrumIndex(Size)

    String getPrimaryIdentifier()
    void setPrimaryIdentifier(String)
```

```
    String getSecondaryIdentifier()
    void setSecondaryIdentifier(String)

    String getCommonName()
    void setCommonName(String)

    String getSumFormula()
    void setSumFormula(String)

    String getInchiString()
    void setInchiString(String)

    String getSMILESString()
    void setSMILESString(String)

    String getPrecursorAdduct()
    void setPrecursorAdduct(String)
```

6.244 Modification

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/Modification.h>" namespace "OpenMS":

  cdef cppclass Modification(SampleTreatment) :
    # wrap-inherits:
    #  SampleTreatment
    Modification()
    Modification(Modification)

    String getReagentName()
    void setReagentName(String & reagent_name)

    double getMass()
    void setMass(double mass)

    Modification_SpecificityType getSpecificityType()
    void setSpecificityType(Modification_SpecificityType & specificity_type)

    String getAffectedAminoAcids()
    void setAffectedAminoAcids(String & affected_amino_acids)

cdef extern from "<OpenMS/METADATA/Modification.h>" namespace\
       "OpenMS::Modification":
```

```
  cdef enum Modification_SpecificityType "OpenMS::Modification::SpecificityType":
    #wrap-attach:
    #  Modification
    AA
    AA_AT_CTERM
    AA_AT_NTERM
    CTERM
    NTERM
    SIZE_OF_SPECIFICITYTYPE
```

6.245 ModificationDefinition

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/ModificationDefinition.h>" namespace "OpenMS":

  cdef cppclass ModificationDefinition "OpenMS::ModificationDefinition":
    # wrap-hash:
    #   getModification().c_str()

    ModificationDefinition()
    ModificationDefinition(ModificationDefinition)
    ModificationDefinition(String &mod)

    bool operator==(ModificationDefinition &rhs)
    bool operator!=(ModificationDefinition &rhs)
    bool operator<(ModificationDefinition &)

    void setFixedModification(bool fixed)
    bool isFixedModification()
    void setMaxOccurences(UInt num)
    UInt getMaxOccurences()
    String getModification()
    void setModification(String &modification)

    void setTermSpecificity(TermSpecificity pos)
    TermSpecificity getTermSpecificity()
```

6.246 ModificationDefinitionsSet

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/ModificationDefinitionsSet.h>" namespace\
       "OpenMS":

  cdef cppclass ModificationDefinitionsSet:

    ModificationDefinitionsSet()
    ModificationDefinitionsSet(ModificationDefinitionsSet rhs)

    ModificationDefinitionsSet(StringList fixed_modifications, StringList\
       variable_modifications)

    void setMaxModifications(Size max_mod)
    Size getMaxModifications()
    Size getNumberOfModifications()
    Size getNumberOfFixedModifications()
    Size getNumberOfVariableModifications()
    void addModification(ModificationDefinition &mod_def)
    void setModifications(libcpp_set[ ModificationDefinition ] &mod_defs)
    void setModifications(String &fixed_modifications, String\
       &variable_modifications)
    void setModifications(StringList &fixed_modifications, StringList\
       &variable_modifications)
    libcpp_set[ ModificationDefinition ] getModifications()
    libcpp_set[ ModificationDefinition ]  getFixedModifications()
    libcpp_set[ ModificationDefinition ]  getVariableModifications()
    libcpp_set[ String ] getModificationNames()
    libcpp_set[ String ] getFixedModificationNames()
    libcpp_set[ String ] getVariableModificationNames()
    bool isCompatible(AASequence &peptide)
```

6.247 ModificationsDB

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/ModificationsDB.h>" namespace "OpenMS":

  cdef cppclass ModificationsDB "OpenMS::ModificationsDB":
    # wrap-manual-memory

    ModificationsDB(ModificationsDB)  #wrap-ignore

    Size getNumberOfModifications()
    ResidueModification getModification(Size index)

    void searchModifications(libcpp_set[ const ResidueModification * ] & mods,
```

```
                String mod_name, String residue,
                TermSpecificity term_spec)

    ResidueModification getModification(String & mod_name, String & residue,\
       TermSpecificity term_spec)

    bool has(String modification)

    Size findModificationIndex(String & mod_name)

    void searchModificationsByDiffMonoMass(libcpp_vector[ String ] & mods, double\
       mass, double max_error,
                        String & residue, TermSpecificity term_spec)

    const ResidueModification* getBestModificationByMonoMass(double mass, double\
       max_error,
                               String residue,
                               TermSpecificity term_spec)

    const ResidueModification* getBestModificationByDiffMonoMass(double mass, double\
       max_error,
                                 String residue, TermSpecificity term_spec)

    void readFromOBOFile(String & filename)
    void readFromUnimodXMLFile(String & filename)
    void getAllSearchModifications(libcpp_vector[ String ] & modifications)

cdef extern from "<OpenMS/CHEMISTRY/ModificationsDB.h>" namespace\
       "OpenMS::ModificationsDB":

  ModificationsDB* getInstance()  # wrap-ignore
```

6.248 ModifiedPeptideGenerator

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/RNPXL/ModifiedPeptideGenerator.h>" namespace\
       "OpenMS":

  cdef cppclass ModifiedPeptideGenerator:

    ModifiedPeptideGenerator()
    ModifiedPeptideGenerator(ModifiedPeptideGenerator)
```

6.249 ModifierRep

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/ModifierRep.h>" namespace "OpenMS":

  cdef cppclass ModifierRep "OpenMS::ModifierRep":
    ModifierRep()
    ModifierRep(ModifierRep)
    void setNumberOfModifications(Size i)
    Size getNumberOfModifications()
    libcpp_vector[ libcpp_vector[ double ] ]  getModificationTable()
    void refreshModificationList(libcpp_map[ double, ptrdiff_t ] & mod_map, char & c)
    Size getMaxModificationMasses()
```

6.250 MorphologicalFilter

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/BASELINE/MorphologicalFilter.h>" namespace\
       "OpenMS":

  cdef cppclass MorphologicalFilter(DefaultParamHandler,ProgressLogger):
    # wrap-inherits:
    #  DefaultParamHandler
    #  ProgressLogger

    MorphologicalFilter()
    void filter(MSSpectrum[Peak1D] & spectrum)
    void filterExperiment(MSExperiment[Peak1D,ChromatogramPeak] & exp)
```

6.251 MsInspectFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/MsInspectFile.h>" namespace "OpenMS":

  cdef cppclass MsInspectFile "OpenMS::MsInspectFile":
    MsInspectFile()
    MsInspectFile(MsInspectFile)  #wrap-ignore
    void load(String & filename, FeatureMap & feature_map)
    void store(String & filename, MSSpectrum[Peak1D] & spectrum)
```

6.252 MultiplexDeltaMasses

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/MultiplexDeltaMasses.h>"\
       namespace "OpenMS":

  cdef cppclass MultiplexDeltaMasses "OpenMS::MultiplexDeltaMasses":
    MultiplexDeltaMasses()
    MultiplexDeltaMasses(MultiplexDeltaMasses)  #wrap-ignore

    MultiplexDeltaMasses(libcpp_vector[ MultiplexDeltaMasses_DeltaMass ] & dm)
    libcpp_vector[ MultiplexDeltaMasses_DeltaMass ]  getDeltaMasses()

cdef extern from "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/MultiplexDeltaMasses.h>"\
       namespace "OpenMS::MultiplexDeltaMasses":

  cdef cppclass MultiplexDeltaMasses_DeltaMass\
       "OpenMS::MultiplexDeltaMasses::DeltaMass":

    MultiplexDeltaMasses_DeltaMass(MultiplexDeltaMasses_DeltaMass)  #wrap-ignore
    MultiplexDeltaMasses_DeltaMass(double dm, String l)
    # DeltaMass(double dm, LabelSet ls);

    double delta_mass
```

6.253 MultiplexDeltaMassesGenerator

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from\
       "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/MultiplexDeltaMassesGenerator.h>" namespace\
       "OpenMS":

  cdef cppclass MultiplexDeltaMassesGenerator(DefaultParamHandler) :
    # wrap-inherits:
    #  DefaultParamHandler
    MultiplexDeltaMassesGenerator()
    MultiplexDeltaMassesGenerator(MultiplexDeltaMassesGenerator)  #wrap-ignore
    MultiplexDeltaMassesGenerator(String labels, int missed_cleavages, libcpp_map[\
       String, double ] label_mass_shift)
```

```
    void generateKnockoutDeltaMasses()
    void printSamplesLabelsList()
    void printDeltaMassesList()
    libcpp_vector[ MultiplexDeltaMasses ] getDeltaMassesList()
    String getLabelShort(String label)
    String getLabelLong(String label)

cdef extern from\
       "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/MultiplexDeltaMassesGenerator.h>" namespace\
       "OpenMS::MultiplexDeltaMassesGenerator":

  cdef cppclass MultiplexDeltaMassesGenerator_Label\
       "OpenMS::MultiplexDeltaMassesGenerator::Label":

    MultiplexDeltaMassesGenerator_Label(MultiplexDeltaMassesGenerator_Label) \
       #wrap-ignore
    MultiplexDeltaMassesGenerator_Label(String sn, String ln, String d, double dm)

    String short_name
    String long_name
    String description
    double delta_mass
```

6.254 MultiplexIsotopicPeakPattern

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from\
       "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/MultiplexIsotopicPeakPattern.h>" namespace\
       "OpenMS":

  cdef cppclass MultiplexIsotopicPeakPattern\
       "OpenMS::MultiplexIsotopicPeakPattern":

    MultiplexIsotopicPeakPattern(MultiplexIsotopicPeakPattern)  #wrap-ignore
    MultiplexIsotopicPeakPattern(int c, int ppp, MultiplexDeltaMasses ms, int msi)

    int getCharge()
    int getPeaksPerPeptide()
    MultiplexDeltaMasses getMassShifts()
    int getMassShiftIndex()
    unsigned getMassShiftCount()
    double getMassShiftAt(int i)
    double getMZShiftAt(int i)
    unsigned getMZShiftCount()
```

6.255 MzDataFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/MzDataFile.h>" namespace "OpenMS":

  cdef cppclass MzDataFile(ProgressLogger):
    # wrap-inherits:
    #   ProgressLogger

    MzDataFile()

    void load(String, MSExperiment[Peak1D, ChromatogramPeak] &) nogil except+
    void store(String, MSExperiment[Peak1D, ChromatogramPeak] &) nogil except+

    PeakFileOptions getOptions()
    void setOptions(PeakFileOptions)

    bool isSemanticallyValid(String & filename, StringList & errors, StringList &\
       warnings)
```

6.256 MzIdentMLFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/MzIdentMLFile.h>" namespace "OpenMS":

  cdef cppclass MzIdentMLFile(ProgressLogger):
    # wrap-inherits:
    #   ProgressLogger

    MzIdentMLFile()

    void load(String filename, Identification & id)
    void load(String filename, libcpp_vector[ProteinIdentification] & poid,\
       libcpp_vector[PeptideIdentification] & peid)
    void store(String filename, Identification & id)
    void store(String filename, libcpp_vector[ProteinIdentification] & poid,\
       libcpp_vector[PeptideIdentification] & peid)
    bool isSemanticallyValid(String filename, StringList errors, StringList\
       warnings)
```

6.257 MzMLFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/MzMLFile.h>" namespace "OpenMS":

  cdef cppclass MzMLFile(ProgressLogger):
    # wrap-inherits:
    #   ProgressLogger

    MzMLFile()

    void load(String, MSExperiment[Peak1D, ChromatogramPeak] &) nogil except+
    void store(String, MSExperiment[Peak1D, ChromatogramPeak] &) nogil except+

    void transform(String, IMSDataConsumer[Peak1D, ChromatogramPeak] *)  #\
       wrap-ignore

    PeakFileOptions getOptions()
    void setOptions(PeakFileOptions)

    bool isSemanticallyValid(String & filename, StringList & errors, StringList &\
       warnings)
```

6.258 MzMLSpectrumDecoder

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/HANDLERS/MzMLSpectrumDecoder.h>" namespace\
       "OpenMS":

  cdef cppclass MzMLSpectrumDecoder "OpenMS::MzMLSpectrumDecoder":
    MzMLSpectrumDecoder()
    MzMLSpectrumDecoder(MzMLSpectrumDecoder)
    void domParseChromatogram(libcpp_string in_, shared_ptr[Chromatogram] & cptr)
    void domParseSpectrum(libcpp_string in_, shared_ptr[Spectrum] & cptr)
    void setSkipXMLChecks(bool only)
```

6.259 MzMLValidator

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/VALIDATORS/MzMLValidator.h>" namespace\
       "OpenMS::Internal":

  cdef cppclass Internal_MzMLValidator "OpenMS::Internal::MzMLValidator":
    Internal_MzMLValidator(Internal_MzMLValidator)  #wrap-ignore
    Internal_MzMLValidator(CVMappings &mapping, ControlledVocabulary &cv)
    # ~MzMLValidator()
```

6.260 MzQuantMLFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/MzQuantMLFile.h>" namespace "OpenMS":

  cdef cppclass MzQuantMLFile:
    MzQuantMLFile()

    void load(String, MSQuantifications &) nogil except+
    void store(String, MSQuantifications &) nogil except+

    bool isSemanticallyValid(String filename,
                 StringList & errors,
                 StringList & warnings)
```

6.261 MzTab

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/MzTab.h>" namespace "OpenMS":

  cdef cppclass MzTab:

    MzTab()
    MzTab(MzTab)  # wrap-ignore
```

6.262 MzTabFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/MzTabFile.h>" namespace "OpenMS":

  cdef cppclass MzTabFile:

    MzTabFile()

    void store(String filename, MzTab & mz_tab)
    void load(String filename, MzTab & mz_tab)
```

6.263 MzXMLFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/MzXMLFile.h>" namespace "OpenMS":

  cdef cppclass MzXMLFile(ProgressLogger):
    # wrap-inherits:
    #   ProgressLogger

    MzXMLFile()

    void load(String, MSExperiment[Peak1D, ChromatogramPeak] &) nogil except+
    void store(String, MSExperiment[Peak1D, ChromatogramPeak] &) nogil except+

    void transform(String, IMSDataConsumer[Peak1D, ChromatogramPeak] *)  #\
       wrap-ignore

    PeakFileOptions getOptions()
    void setOptions(PeakFileOptions)
```

6.264 NLargest

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/TRANSFORMERS/NLargest.h>" namespace "OpenMS":

  cdef cppclass NLargest(DefaultParamHandler):
    # wrap-inherits:
```

```
    #  DefaultParamHandler

    NLargest()
    NLargest(NLargest)  #wrap-ignore

    void filterSpectrum(MSSpectrum[Peak1D] & spec)
    void filterPeakSpectrum(MSSpectrum[Peak1D] & spec)
    void filterPeakMap(MSExperiment[Peak1D, ChromatogramPeak] & exp)
```

6.265 NeutralLossDiffFilter

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/TRANSFORMERS/NeutralLossDiffFilter.h>" namespace\
       "OpenMS":

  cdef cppclass NeutralLossDiffFilter(FilterFunctor) :
    # wrap-inherits:
    #  FilterFunctor
    NeutralLossDiffFilter()
    NeutralLossDiffFilter(NeutralLossDiffFilter)
    double apply(MSSpectrum[Peak1D] & )
    # POINTER # FilterFunctor * create()
    String getProductName()
```

6.266 NeutralLossMarker

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/TRANSFORMERS/NeutralLossMarker.h>" namespace\
       "OpenMS":

  cdef cppclass NeutralLossMarker(PeakMarker) :
    # wrap-inherits:
    #  PeakMarker
    NeutralLossMarker()
    NeutralLossMarker(NeutralLossMarker)
    void apply(libcpp_map[ double, bool ] & , MSSpectrum[Peak1D] & )
    PeakMarker * create()  # wrap-ignore
    # String getProductName()
```

6.267 NonNegativeLeastSquaresSolver

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/MATH/MISC/NonNegativeLeastSquaresSolver.h>" namespace\
       "OpenMS":

  cdef cppclass NonNegativeLeastSquaresSolver\
       "OpenMS::NonNegativeLeastSquaresSolver":
    NonNegativeLeastSquaresSolver(NonNegativeLeastSquaresSolver)  #wrap-ignore
    Int solve(Matrix[ double ] & A, Matrix[ double ] & b, Matrix[ double ] & x)

cdef extern from "<OpenMS/MATH/MISC/NonNegativeLeastSquaresSolver.h>" namespace\
       "OpenMS::NonNegativeLeastSquaresSolver":
  cdef enum RETURN_STATUS "OpenMS::NonNegativeLeastSquaresSolver::RETURN_STATUS":
    #wrap-attach:
    #  NonNegativeLeastSquaresSolver
    SOLVED
    ITERATION_EXCEEDED
```

6.268 Normalizer

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/TRANSFORMERS/Normalizer.h>" namespace "OpenMS":

  cdef cppclass Normalizer(DefaultParamHandler):
    # wrap-inherits:
    #  DefaultParamHandler

    Normalizer()
    Normalizer(Normalizer)  #wrap-ignore

    void filterSpectrum(MSSpectrum[Peak1D] & spec)
    void filterPeakSpectrum(MSSpectrum[Peak1D] & spec)
    void filterPeakMap(MSExperiment[Peak1D, ChromatogramPeak] & exp)
```

6.269 OMSSACSVFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/OMSSACSVFile.h>" namespace "OpenMS":

  cdef cppclass OMSSACSVFile "OpenMS::OMSSACSVFile":
    OMSSACSVFile()
    OMSSACSVFile(OMSSACSVFile)  #wrap-ignore

    void load(String & filename,
          ProteinIdentification & protein_identification,
          libcpp_vector[ PeptideIdentification ] & id_data)
```

6.270 OMSSAXMLFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/OMSSAXMLFile.h>" namespace "OpenMS":

  cdef cppclass OMSSAXMLFile(XMLFile) :
    # wrap-inherits:
    #  XMLFile

    OMSSAXMLFile()
    OMSSAXMLFile(OMSSAXMLFile)  #wrap-ignore

    void load(String & filename,
          ProteinIdentification & protein_identification,
          libcpp_vector[ PeptideIdentification ] & id_data,
          bool load_proteins,
          bool load_empty_hits)

    void setModificationDefinitionsSet(ModificationDefinitionsSet rhs)
```

6.271 OSChromatogramMeta

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from\
       "<OpenMS/ANALYSIS/OPENSWATH/OPENSWATHALGO/DATAACCESS/DataStructures.h>" namespace\
       "OpenSwath":

  cdef cppclass OSChromatogramMeta "OpenSwath::OSChromatogramMeta":

    OSChromatogramMeta()
```

```
    OSChromatogramMeta(OSChromatogramMeta)  #wrap-ignore

    size_t index
    libcpp_string id
```

6.272 OSSpectrumMeta

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from\
       "<OpenMS/ANALYSIS/OPENSWATH/OPENSWATHALGO/DATAACCESS/DataStructures.h>" namespace\
       "OpenSwath":

  cdef cppclass OSSpectrumMeta "OpenSwath::OSSpectrumMeta":

    OSSpectrumMeta()
    OSSpectrumMeta(OSSpectrumMeta)  #wrap-ignore

    size_t index
    libcpp_string id
    double RT
    int ms_level
```

6.273 OfflinePrecursorIonSelection

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/TARGETED/OfflinePrecursorIonSelection.h>"\
       namespace "OpenMS":

  cdef cppclass OfflinePrecursorIonSelection(DefaultParamHandler) :
    # wrap-inherits:
    #  DefaultParamHandler
    OfflinePrecursorIonSelection()
    OfflinePrecursorIonSelection(OfflinePrecursorIonSelection)  #wrap-ignore

    void makePrecursorSelectionForKnownLCMSMap(FeatureMap & features,
                           MSExperiment[ Peak1D, ChromatogramPeak ] & experiment,
                           MSExperiment[ Peak1D, ChromatogramPeak ] & ms2,
                           libcpp_set[ int ] & charges_set, bool
                           feature_based)
```

```
    void getMassRanges(FeatureMap & features,
               MSExperiment[ Peak1D, ChromatogramPeak ] & experiment,
               libcpp_vector[ libcpp_vector[ libcpp_pair[ Size, Size ] ] ] &\
       indices)  # wrap-ignore

    void createProteinSequenceBasedLPInclusionList(String include_, String\
       rt_model_file, String pt_model_file, FeatureMap & precursors)
    void setLPSolver(SOLVER solver)
    SOLVER getLPSolver()
```

6.274 OnDiscMSExperiment

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/KERNEL/OnDiscMSExperiment.h>" namespace "OpenMS":

  cdef cppclass OnDiscMSExperiment[PeakT, ChromoPeakT](ExperimentalSettings):
    # wrap-instances:
    #   OnDiscMSExperiment := OnDiscMSExperiment[Peak1D, ChromatogramPeak]

    OnDiscMSExperiment()
    OnDiscMSExperiment(OnDiscMSExperiment &)

    bool openFile(String filename)
    Size getNrSpectra()
    Size getNrChromatograms()

    shared_ptr[const ExperimentalSettings] getExperimentalSettings()

    MSSpectrum[PeakT] getSpectrum(Size id)
    MSChromatogram[ChromoPeakT] getChromatogram(Size id)

    shared_ptr[Spectrum] getSpectrumById(int id_)
    shared_ptr[Chromatogram] getChromatogramById(int id_)

    void setSkipXMLChecks(bool skip)
```

6.275 OpenSwathDataAccessHelper

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/OPENSWATH/DATAACCESS/DataAccessHelper.h>"\
```

```
      namespace "OpenMS":

  cdef cppclass OpenSwathDataAccessHelper:

    # OpenSwathDataAccessHelper(OpenSwathDataAccessHelper)  # wrap-ignore

    void convertToOpenMSSpectrum(shared_ptr[OSSpectrum] sptr, MSSpectrum[Peak1D] &\
      spectrum)

    shared_ptr[OSSpectrum] convertToSpectrumPtr(MSSpectrum[Peak1D] spectrum)  #\
      wrap-ignore

    void convertToOpenMSChromatogram(MSChromatogram[ChromatogramPeak] &\
      chromatogram, shared_ptr[OSChromatogram] cptr)

    void convertTargetedExp(TargetedExperiment & transition_exp_,\
      LightTargetedExperiment & transition_exp)

    void convertPeptideToAASequence(LightCompound & peptide, AASequence &\
      aa_sequence)

    void convertTargetedCompound(Peptide pep, LightCompound & p)
```

6.276 OpenSwathDataStructures

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from\
       "<OpenMS/ANALYSIS/OPENSWATH/OPENSWATHALGO/DATAACCESS/DataStructures.h>" namespace\
       "OpenSwath":

  cdef cppclass OSBinaryDataArray:
    OSBinaryDataArray()
    OSBinaryDataArray(OSBinaryDataArray)
    libcpp_vector[double] data

  ctypedef shared_ptr[OSBinaryDataArray] OSBinaryDataArrayPtr

  cdef cppclass OSSpectrum:
    OSSpectrum()
    OSSpectrum(OSSpectrum)
    OSBinaryDataArrayPtr getMZArray() #wrap-ignore
    OSBinaryDataArrayPtr getIntensityArray() #wrap-ignore
    void setMZArray(OSBinaryDataArrayPtr data) #wrap-ignore
    void setIntensityArray(OSBinaryDataArrayPtr data) #wrap-ignore
```

```
  ctypedef shared_ptr[OSSpectrum] OSSpectrumPtr

  cdef cppclass OSChromatogram:
    OSChromatogram()
    OSChromatogram(OSChromatogram)
    OSBinaryDataArrayPtr getTimeArray() #wrap-ignore
    OSBinaryDataArrayPtr getIntensityArray() #wrap-ignore
    void setTimeArray(OSBinaryDataArrayPtr data) #wrap-ignore
    void setIntensityArray(OSBinaryDataArrayPtr data) #wrap-ignore

  ctypedef shared_ptr[OSChromatogram] OSChromatogramPtr
```

6.277 OpenSwathHelper

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/OPENSWATH/OpenSwathHelper.h>" namespace\
       "OpenMS":

  cdef cppclass OpenSwathHelper:

    bool checkSwathMapAndSelectTransitions(
                MSExperiment[Peak1D, ChromatogramPeak] & exp,
                TargetedExperiment & targeted_exp,
                TargetedExperiment & transition_exp_used,
                double min_upper_edge_dist
                )

    libcpp_pair[double, double] estimateRTRange(LightTargetedExperiment exp)
```

6.278 OpenSwathScoring

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/OPENSWATH/OpenSwathScoring.h>" namespace\
       "OpenMS":

  cdef cppclass OpenSwathScoring:

    OpenSwathScoring()
    OpenSwathScoring(OpenSwathScoring)  # wrap-ignore
```

```
  void initialize(double rt_normalization_factor_,
    int add_up_spectra_, double spacing_for_spectra_resampling_,
    OpenSwath_Scores_Usage & su_)

  void getNormalized_library_intensities_(libcpp_vector[LightTransition]\
     transitions,
                    libcpp_vector[double] normalized_library_intensity)

  shared_ptr[OSSpectrum] getAddedSpectra_(shared_ptr[ SpectrumAccessOpenMS ]\
     swath_map,
                    double RT, int nr_spectra_to_add)

cdef cppclass OpenSwath_Scores_Usage:

  OpenSwath_Scores_Usage()
  OpenSwath_Scores_Usage(OpenSwath_Scores_Usage)  # wrap-ignore

  bool use_coelution_score_
  bool use_shape_score_
  bool use_rt_score_
  bool use_library_score_
  bool use_elution_model_score_
  bool use_intensity_score_
  bool use_total_xic_score_
  bool use_nr_peaks_score_
  bool use_sn_score_
  bool use_dia_scores_
  bool use_ms1_correlation
  bool use_ms1_fullscan

cdef cppclass OpenSwath_Scores:

  OpenSwath_Scores()
  OpenSwath_Scores(OpenSwath_Scores)  # wrap-ignore

  double get_quick_lda_score(double library_corr_, double
                library_norm_manhattan_, double
                norm_rt_score_, double
                xcorr_coelution_score_, double
                xcorr_shape_score_, double log_sn_score_)

  double calculate_lda_prescore(OpenSwath_Scores scores)
  double calculate_swath_lda_prescore(OpenSwath_Scores scores)

  double elution_model_fit_score
  double library_corr
  double library_norm_manhattan
```

```
    double library_rootmeansquare
    double library_sangle
    double norm_rt_score
    double isotope_correlation
    double isotope_overlap
    double massdev_score
    double xcorr_coelution_score
    double xcorr_shape_score
    double yseries_score
    double bseries_score
    double log_sn_score

    double weighted_coelution_score
    double weighted_xcorr_shape
    double weighted_massdev_score

    double xcorr_ms1_coelution_score
    double xcorr_ms1_shape_score
    double ms1_ppm_score
    double ms1_isotope_correlation
    double ms1_isotope_overlap

    double library_manhattan
    double library_dotprod
    double intensity
    double total_xic
    double nr_peaks
    double sn_ratio

    double rt_difference
    double normalized_experimental_rt
    double raw_rt_score

    double dotprod_score_dia
    double manhatt_score_dia
```

6.279 OptimizePeakDeconvolution

→ Link to OpenMS documentation

Wrapped functions in Python:

```
cdef extern from "<OpenMS/TRANSFORMATIONS/RAW2PEAK/OptimizePeakDeconvolution.h>"\
       namespace "OpenMS":

  cdef cppclass OptimizePeakDeconvolution(DefaultParamHandler) :
    # wrap-inherits:
```

```
    #  DefaultParamHandler
    OptimizePeakDeconvolution()
    OptimizePeakDeconvolution(OptimizePeakDeconvolution)
    PenaltyFactorsIntensity  getPenalties()
    void setPenalties(PenaltyFactorsIntensity & penalties)
    Int getCharge()
    void setCharge(Int charge)
    bool optimize(libcpp_vector[ PeakShape ] & peaks, OptimizePeakDeconvolution_Data\
       & data)
    Size getNumberOfPeaks_(Int charge, libcpp_vector[ PeakShape ] & temp_shapes,\
       OptimizePeakDeconvolution_Data & data)

cdef extern from "<OpenMS/TRANSFORMATIONS/RAW2PEAK/OptimizePeakDeconvolution.h>"\
       namespace "OpenMS::OptimizationFunctions":

  cdef cppclass PenaltyFactorsIntensity(OptimizationFunctions_PenaltyFactors):
    # wrap-inherits:
    #  OptimizationFunctions_PenaltyFactors
    PenaltyFactorsIntensity()
    PenaltyFactorsIntensity(PenaltyFactorsIntensity)
    double height

cdef extern from "<OpenMS/TRANSFORMATIONS/RAW2PEAK/OptimizePeakDeconvolution.h>"\
       namespace "OpenMS::OptimizePeakDeconvolution":

  cdef cppclass OptimizePeakDeconvolution_Data\
       "OpenMS::OptimizePeakDeconvolution::Data":
    OptimizePeakDeconvolution_Data()
    OptimizePeakDeconvolution_Data(OptimizePeakDeconvolution_Data)  # no-wrap

    libcpp_vector[PeakShape] peaks
    libcpp_vector[double] positions
    libcpp_vector[double] signal
    PenaltyFactorsIntensity penalties
    Int charge
```

6.280 OptimizePick

→ Link to OpenMS documentation

Wrapped functions in Python:

```
cdef extern from "<OpenMS/TRANSFORMATIONS/RAW2PEAK/OptimizePick.h>" namespace\
       "OpenMS":

  cdef cppclass OptimizePick "OpenMS::OptimizePick":
    OptimizePick()
```

```
    OptimizePick(OptimizePick)  #wrap-ignore
    OptimizePick(OptimizationFunctions_PenaltyFactors penalties_, int\
       max_iteration_)
    OptimizationFunctions_PenaltyFactors getPenalties()
    void setPenalties(OptimizationFunctions_PenaltyFactors penalties)
    unsigned int  getNumberIterations()
    void setNumberIterations(int max_iteration)
    # void optimize(libcpp_vector[ PeakShape ] & peaks, OptimizePick_Data & data)

cdef extern from "<OpenMS/TRANSFORMATIONS/RAW2PEAK/OptimizePick.h>" namespace\
       "OpenMS::OptimizationFunctions":

  cdef cppclass OptimizationFunctions_PenaltyFactors\
       "OpenMS::OptimizationFunctions::PenaltyFactors":
    OptimizationFunctions_PenaltyFactors()
    OptimizationFunctions_PenaltyFactors(OptimizationFunctions_PenaltyFactors)
    double pos
    double lWidth
    double rWidth

cdef extern from "<OpenMS/TRANSFORMATIONS/RAW2PEAK/OptimizePick.h>" namespace\
       "OpenMS::OptimizePick":

  cdef cppclass OptimizePick_Data "OpenMS::OptimizePick::Data":
    OptimizePick_Data(OptimizePick_Data)  #wrap-ignore
    libcpp_vector[ double ] positions
    libcpp_vector[ double ] signal
    # TODO STL attribute
    # libcpp_vector[ PeakShape ] peaks
    # NAMESPACE # OptimizationFunctions::PenaltyFactors penalties
```

6.281 PSLPFormulation

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/TARGETED/PSLPFormulation.h>" namespace "OpenMS":

  cdef cppclass PSLPFormulation(DefaultParamHandler) :
    # wrap-inherits:
    #  DefaultParamHandler

    PSLPFormulation()
    PSLPFormulation(PSLPFormulation)  #wrap-ignore

    void createAndSolveILPForKnownLCMSMapFeatureBased(
```

```
  FeatureMap & features,
  MSExperiment[ Peak1D, ChromatogramPeak ] & experiment,
  libcpp_vector[ IndexTriple ] & variable_indices,
  libcpp_vector[ libcpp_vector[ libcpp_pair[ size_t, size_t ] ] ] & mass_ranges,
  libcpp_set[ int ] & charges_set,
  UInt ms2_spectra_per_rt_bin,
  libcpp_vector[ int ] & solution_indices)

void createAndSolveILPForInclusionListCreation(
  PrecursorIonSelectionPreprocessing & preprocessing,
  UInt ms2_spectra_per_rt_bin, UInt max_list_size, FeatureMap & precursors, bool\
   solve_ILP)

void createAndSolveCombinedLPForKnownLCMSMapFeatureBased(FeatureMap & features,\
   MSExperiment[ Peak1D, ChromatogramPeak ] & experiment,
  libcpp_vector[ IndexTriple ] & variable_indices, libcpp_vector[ int ] &\
   solution_indices,
  libcpp_vector[ libcpp_vector[ libcpp_pair[ size_t, size_t ] ] ] & mass_ranges,
  libcpp_set[ Int ] & charges_set, UInt ms2_spectra_per_rt_bin, Size step_size,\
   bool sequential_order)  # wrap-ignore

void updateStepSizeConstraint(Size iteration, UInt step_size)

void updateFeatureILPVariables(
  FeatureMap & new_features,
  libcpp_vector[ IndexTriple ] & variable_indices,
  libcpp_map[ Size, libcpp_vector[ String ] ] & feature_constraints_map)  #\
   wrap-ignore

void updateRTConstraintsForSequentialILP(Size & rt_index, UInt\
   ms2_spectra_per_rt_bin, Size max_rt_index)

void updateCombinedILP(FeatureMap & features,
             PrecursorIonSelectionPreprocessing & preprocessed_db,
             libcpp_vector[ IndexTriple ] & variable_indices,
             libcpp_vector[ String ] & new_protein_accs,
             libcpp_vector[ String ] & protein_accs,
             PSProteinInference & prot_inference,
             Size & variable_counter,
             libcpp_map[ String, libcpp_vector[ size_t ] ] &\
   protein_feature_map,
             Feature & new_feature, libcpp_map[ String, Size ] &\
   protein_variable_index_map,
             libcpp_map[ String, libcpp_set[ String ] ] & prot_id_counter)  #\
   wrap-ignore

void solveILP(libcpp_vector[ int ] & solution_indices)
```

```
    void setLPSolver(SOLVER solver)

    SOLVER getLPSolver()

cdef extern from "<OpenMS/ANALYSIS/TARGETED/PSLPFormulation.h>" namespace\
       "OpenMS::PSLPFormulation":

  cdef cppclass IndexTriple "OpenMS::PSLPFormulation::IndexTriple":
    IndexTriple()
    IndexTriple(IndexTriple)  #wrap-ignore
    Size feature
    Int scan
    Size variable
    double rt_probability
    double signal_weight
    String prot_acc
```

6.282 PSProteinInference

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/TARGETED/PSProteinInference.h>" namespace\
       "OpenMS":

  cdef cppclass PSProteinInference "OpenMS::PSProteinInference":
    PSProteinInference()
    PSProteinInference(PSProteinInference)  #wrap-ignore
    Size findMinimalProteinList(libcpp_vector[ PeptideIdentification ] &\
       peptide_ids)
    void calculateProteinProbabilities(libcpp_vector[ PeptideIdentification ] & ids)
    double getProteinProbability(String & acc)
    bool isProteinInMinimalList(String & acc)
    Int getNumberOfProtIds(double protein_id_threshold)
    # TODO nested STL
    # Int getNumberOfProtIdsPeptideRule(Int min_peptides, libcpp_map[ String,\
       libcpp_set[ String ] ] & prot_id_counter)
    void setSolver(SOLVER solver)
    SOLVER getSolver()
```

6.283 Param

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/DATASTRUCTURES/Param.h>" namespace "OpenMS":

  cdef cppclass Param:
     Param()
     Param(Param)
     bool operator==(Param)

     void setValue(String key, DataValue val, String desc, StringList tags)
     void setValue(String key, DataValue val, String desc)
     DataValue getValue(String key)
     ParamEntry getEntry(String)
     int exists(String key)

     void addTag(String key, String tag)
     void addTags(String key, StringList tags)
     int hasTag(String key, String tag)
     StringList getTags(String key)
     void clearTags(String key)

     libcpp_string getDescription(String key)
     void setSectionDescription(String key, String desc)
     libcpp_string getSectionDescription(String key)

     Size size()
     bool empty()

     void clear()
     void insert(String prefix, Param param)

     void remove(String key)
     void removeAll(String prefix)

     Param copy(String prefix, bool)
     Param copy(String prefix)

     void update(Param p_old, bool add_unknow)  # wrap-ignore
     void update(Param p_old)  # wrap-ignore

     void merge(Param toMerge)

     void setDefaults(Param defaults, String previx, bool showMessage)
     void setDefaults(Param defaults, String previx)
     void setDefaults(Param defaults)

     void checkDefaults(String name, Param defaults, String prefix)
     void checkDefaults(String name, Param defaults)

     void setValidStrings(String key, libcpp_vector[String] strings)
```

```
    void setMinInt(String key, int min)
    void setMaxInt(String key, int max)
    void setMinFloat(String key, double min)
    void setMaxFloat(String key, double max)

    #void parseCommandLine(int argc, char ** argv, String prefix) # wrap-ignore
    #void parseCommandLine(int argc, char ** argv) # wrap-ignore

    ParamIterator begin()  # wrap-ignore
    ParamIterator end()    # wrap-ignore

cdef extern from "<OpenMS/DATASTRUCTURES/Param.h>" namespace "OpenMS::Param":

  cdef cppclass ParamIterator:
    # wrap-ignore
    ParamIterator operator++()
    ParamIterator operator--()
    String getName()
    int operator==(ParamIterator)
    int operator!=(ParamIterator)
    int operator<(ParamIterator)
    int operator>(ParamIterator)
    int operator<=(ParamIterator)
    int operator>=(ParamIterator)
    # Returns the traceback of the opened and closed sections
    libcpp_vector[TraceInfo] getTrace()

cdef extern from "<OpenMS/DATASTRUCTURES/Param.h>" namespace\
       "OpenMS::Param::ParamIterator":

  cdef cppclass TraceInfo:

    TraceInfo(String n, String d, bool o)
    TraceInfo(TraceInfo)

    String name
    String description
    bool opened
```

6.284 ParamEntry

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/DATASTRUCTURES/Param.h>" namespace "OpenMS::Param":

  cdef cppclass ParamEntry:
```

```
    String name
    String description
    DataValue value
    libcpp_set[String] tags
    libcpp_vector[String] valid_strings
    double  max_float
    double  min_float
    Int  max_int
    Int  min_int

    ParamEntry()
    ParamEntry(ParamEntry)
    ParamEntry(String n, DataValue v, String d, StringList t)
    ParamEntry(String n, DataValue v, String d)

    bool isValid(String &message)
    bool operator==(ParamEntry)
```

6.285 ParamNode

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/DATASTRUCTURES/Param.h>" namespace "OpenMS::Param":

  cdef cppclass ParamNode "OpenMS::Param::ParamNode":
    ParamNode()
    ParamNode(ParamNode)  #wrap-ignore
    String name
    String description
    libcpp_vector[ ParamEntry ] entries
    libcpp_vector[ ParamNode ] nodes
    ParamNode(String & n, String & d)
    bool operator==(ParamNode & rhs)
    # EntryIterator findEntry(String & name)
    # NodeIterator findNode(String & name)
    ParamNode * findParentOf(String & name)
    ParamEntry * findEntryRecursive(String & name)
    void insert(ParamNode & node, String & prefix)
    void insert(ParamEntry & entry, String & prefix)
    Size size()
    String suffix(String & key)
```

6.286 ParamXMLFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/ParamXMLFile.h>" namespace "OpenMS":

  cdef cppclass ParamXMLFile:

    ParamXMLFile()
    void load(String, Param &) nogil except+
    void store(String, Param &) nogil except+
```

6.287 ParentPeakMower

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/TRANSFORMERS/ParentPeakMower.h>" namespace\
       "OpenMS":

  cdef cppclass ParentPeakMower(DefaultParamHandler):
    # wrap-inherits:
    #  DefaultParamHandler

    ParentPeakMower()
    ParentPeakMower(ParentPeakMower)  #wrap-ignore

    void filterSpectrum(MSSpectrum[Peak1D] & spec)
    void filterPeakSpectrum(MSSpectrum[Peak1D] & spec)
    void filterPeakMap(MSExperiment[Peak1D, ChromatogramPeak] & exp)
```

6.288 Peak1D

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/KERNEL/Peak1D.h>" namespace "OpenMS":

  cdef cppclass Peak1D:
    Peak1D()
    Peak1D(Peak1D &)

    float getIntensity()
    double getMZ()
```

```
    void setMZ(double)
    void setIntensity(float)
    bool operator==(Peak1D)
    bool operator!=(Peak1D)
    double getPos()
    void setPos(double pos)
    # DPosition1 getPosition()  # wrap-ignore
    # void setPosition(DPosition1 position)  # wrap-ignore
```

6.289 Peak2D

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/KERNEL/Peak2D.h>" namespace "OpenMS":

  cdef cppclass Peak2D:
    Peak2D()
    Peak2D(Peak2D &)
    float getIntensity()
    double getMZ()
    double getRT()
    void setMZ(double)
    void setRT(double)
    void setIntensity(float)
    bool operator==(Peak2D)
    bool operator!=(Peak2D)
cdef extern from "<OpenMS/KERNEL/Peak2D.h>" namespace "OpenMS::Peak2D":

  cdef enum DimensionDescription "OpenMS::Peak2D::DimensionDescription":
    RT
    MZ
    DIMENSION
```

6.290 PeakFileOptions

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/OPTIONS/PeakFileOptions.h>" namespace "OpenMS":

  cdef cppclass PeakFileOptions:

    PeakFileOptions()
    PeakFileOptions(PeakFileOptions)
```

```
void setMetadataOnly(bool)
bool getMetadataOnly()

void setWriteSupplementalData(bool)
bool getWriteSupplementalData()

void setMSLevels(libcpp_vector[int] levels)
void addMSLevel(Int level)
void clearMSLevels()
bool hasMSLevels()
bool containsMSLevel(int level)
libcpp_vector[int] getMSLevels()

void setCompression(bool)
bool getCompression()

void setMz32Bit(bool mz_32_bit)
bool getMz32Bit()
void setIntensity32Bit(bool int_32_bit)
bool getIntensity32Bit()

void setRTRange(DRange1 & range_)
bool hasRTRange()
DRange1 getRTRange()
void setMZRange(DRange1 & range_)
bool hasMZRange()
DRange1 getMZRange()
void setIntensityRange(DRange1 & range_)
bool hasIntensityRange()
DRange1 getIntensityRange()

Size getMaxDataPoolSize()
void setMaxDataPoolSize(Size s)

void setSortSpectraByMZ(bool doSort)
bool getSortSpectraByMZ()
void setSortChromatogramsByRT(bool doSort)
bool getSortChromatogramsByRT()

void setSizeOnly(bool only)
bool getSizeOnly()

void setFillData(bool only)
bool getFillData()

void setSkipXMLChecks(bool only)
bool getSkipXMLChecks()
```

```
    bool getWriteIndex()
    void setWriteIndex(bool write_index)

    NumpressConfig getNumpressConfigurationMassTime()
    void setNumpressConfigurationMassTime(NumpressConfig config)

    NumpressConfig getNumpressConfigurationIntensity()
    void setNumpressConfigurationIntensity(NumpressConfig config)
```

6.291 PeakIndex

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/KERNEL/PeakIndex.h>" namespace "OpenMS":

  cdef cppclass PeakIndex "OpenMS::PeakIndex":
    PeakIndex()
    PeakIndex(PeakIndex)  #wrap-ignore
    Size peak
    Size spectrum
    PeakIndex(Size peak)
    PeakIndex(Size spectrum, Size peak)
    bool isValid()
    void clear()
    Feature getFeature(FeatureMap & map_)
    Peak1D getPeak(MSExperiment[Peak1D, ChromatogramPeak] & map_)
    MSSpectrum[Peak1D] getSpectrum(MSExperiment[Peak1D, ChromatogramPeak] & map_)
    bool operator==(PeakIndex & rhs)
    bool operator!=(PeakIndex & rhs)
```

6.292 PeakIntensityPredictor

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/PIP/PeakIntensityPredictor.h>" namespace\
       "OpenMS":

  cdef cppclass PeakIntensityPredictor "OpenMS::PeakIntensityPredictor":
    PeakIntensityPredictor()
    PeakIntensityPredictor(PeakIntensityPredictor)  #wrap-ignore
    double predict(AASequence & sequence)
```

```
    double predict(AASequence & sequence, libcpp_vector[ double ] & add_info)
    libcpp_vector[ double ] predict(libcpp_vector[ AASequence ] & sequences)
    libcpp_vector[ double ] predict(libcpp_vector[ AASequence ] & sequences,\
       libcpp_vector[ libcpp_vector[ double ] ] & add_info)
```

6.293 PeakMarker

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/TRANSFORMERS/PeakMarker.h>" namespace "OpenMS":

  cdef cppclass PeakMarker(DefaultParamHandler) :
    # wrap-inherits:
    #  DefaultParamHandler
    PeakMarker()
    PeakMarker(PeakMarker)
    # see child classes
    # void apply(libcpp_map[ double, bool ] & , MSSpectrum[Peak1D] & )
    String getProductName()
```

6.294 PeakPickerCWT

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/TRANSFORMATIONS/RAW2PEAK/PeakPickerCWT.h>" namespace\
       "OpenMS":

  cdef cppclass PeakPickerCWT(DefaultParamHandler, ProgressLogger):
    # wrap-inherits:
    #  DefaultParamHandler
    #  ProgressLogger

    PeakPickerCWT()
    PeakPickerCWT(PeakPickerCWT)    #wrap-ignore

    void pick(MSSpectrum[Peak1D] & input, MSSpectrum[Peak1D] & output)
    void pickExperiment(MSExperiment[Peak1D, ChromatogramPeak] & input,\
       MSExperiment[Peak1D, ChromatogramPeak] & output)
    double estimatePeakWidth(MSExperiment[Peak1D, ChromatogramPeak] & input)
```

6.295 PeakPickerHiRes

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/TRANSFORMATIONS/RAW2PEAK/PeakPickerHiRes.h>" namespace\
       "OpenMS":

  cdef cppclass PeakPickerHiRes(DefaultParamHandler, ProgressLogger):
    # wrap-inherits:
    #  DefaultParamHandler
    #  ProgressLogger

    PeakPickerHiRes()
    PeakPickerHiRes(PeakPickerHiRes)    #wrap-ignore

    void pick(MSSpectrum[Peak1D] & input,
          MSSpectrum[Peak1D] & output
         )

    void pickExperiment(MSExperiment[Peak1D, ChromatogramPeak] & input,
              MSExperiment[Peak1D, ChromatogramPeak] & output
               )

cdef extern from "<OpenMS/TRANSFORMATIONS/RAW2PEAK/PeakPickerHiRes.h>" namespace\
       "OpenMS::PeakPickerHiRes":

  cdef cppclass PeakBoundary "OpenMS::PeakPickerHiRes::PeakBoundary":
    PeakBoundary()
    PeakBoundary(PeakBoundary)  #wrap-ignore
    double mz_min
    double mz_max
```

6.296 PeakPickerIterative

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/TRANSFORMATIONS/RAW2PEAK/PeakPickerIterative.h>" namespace\
       "OpenMS":

  cdef cppclass PeakPickerIterative(DefaultParamHandler, ProgressLogger):
    # wrap-inherits:
    #  DefaultParamHandler
    #  ProgressLogger

    PeakPickerIterative()
```

```
    PeakPickerIterative(PeakPickerIterative)  #wrap-ignore

    void pick(MSSpectrum[Peak1D] & input,
          MSSpectrum[Peak1D] & output
         )

    void pickExperiment(MSExperiment[Peak1D, ChromatogramPeak] & input,
              MSExperiment[Peak1D, ChromatogramPeak] & output
               )

cdef extern from "<OpenMS/TRANSFORMATIONS/RAW2PEAK/PeakPickerIterative.h>" namespace\
       "OpenMS":

  cdef cppclass PeakCandidate:
    PeakCandidate()
    PeakCandidate(PeakCandidate)
    int index
    double peak_apex_intensity

    double integrated_intensity
    double leftWidth
    double rightWidth
    float mz
```

6.297 PeakPickerMRM

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/OPENSWATH/PeakPickerMRM.h>" namespace "OpenMS":

  cdef cppclass PeakPickerMRM(DefaultParamHandler) :
    # wrap-inherits:
    #  DefaultParamHandler
    PeakPickerMRM()
    PeakPickerMRM(PeakPickerMRM)  #wrap-ignore

    void pickChromatogram(MSSpectrum[ChromatogramPeak] & chromatogram,\
       MSSpectrum[ChromatogramPeak] & picked_chrom)
```

6.298 PeakPickerMaxima

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/TRANSFORMATIONS/RAW2PEAK/PeakPickerMaxima.h>" namespace\
       "OpenMS":

  cdef cppclass PeakPickerMaxima:

    PeakPickerMaxima()  # wrap-ignore
    PeakPickerMaxima(PeakPickerMaxima &)
    PeakPickerMaxima(double signal_to_noise, double spacing_difference, double\
       sn_window_length)

    void findMaxima(libcpp_vector[double] mz_array, libcpp_vector[double]
            int_array, libcpp_vector[PeakCandidate]& pc)

    void pick(libcpp_vector[double] mz_array, libcpp_vector[double]
            int_array, libcpp_vector[PeakCandidate]& pc)

cdef extern from "<OpenMS/TRANSFORMATIONS/RAW2PEAK/PeakPickerMaxima.h>" namespace\
       "OpenMS::PeakPickerMaxima":

  cdef cppclass PeakCandidate "OpenMS::PeakPickerMaxima::PeakCandidate":

    PeakCandidate()
    PeakCandidate(PeakCandidate &)
    int pos
    int left_boundary
    int right_boundary
    double mz_max
    double int_max
```

6.299 PeakPickerSH

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/TRANSFORMATIONS/RAW2PEAK/PeakPickerSH.h>" namespace\
       "OpenMS":

  cdef cppclass PeakPickerSH(DefaultParamHandler,ProgressLogger) :
    # wrap-inherits:
    #  DefaultParamHandler
    #  ProgressLogger
    PeakPickerSH()
    PeakPickerSH(PeakPickerSH)  #wrap-ignore
    void pick(MSSpectrum[ Peak1D ] & input_, MSSpectrum[ Peak1D ] & output, float\
       fWindowWidth)
    void pickExperiment(MSExperiment[Peak1D, ChromatogramPeak] & input_,\
       MSExperiment[Peak1D, ChromatogramPeak] & output)
```

6.300 PeakShape

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/TRANSFORMATIONS/RAW2PEAK/PeakShape.h>" namespace\
       "OpenMS":

  cdef cppclass PeakShape "OpenMS::PeakShape":
    PeakShape()
    PeakShape(PeakShape)
    double height
    double mz_position
    double left_width
    double right_width
    double area
    double r_value
    double signal_to_noise
    PeakShape_Type type
    PeakShape(double height_, double mz_position_, double left_width_, double\
       right_width_, double area_, PeakShape_Type type_)
    bool operator==(PeakShape & rhs)
    bool operator!=(PeakShape & rhs)
    double getSymmetricMeasure()
    double getFWHM()
    bool iteratorsSet()
cdef extern from "<OpenMS/TRANSFORMATIONS/RAW2PEAK/PeakShape.h>" namespace\
       "OpenMS::PeakShape":
  cdef enum PeakShape_Type "OpenMS::PeakShape::Type":
    #wrap-attach:
    #  PeakShape
    LORENTZ_PEAK
    SECH_PEAK
    UNDEFINED
```

6.301 PeakTypeEstimator

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/PeakTypeEstimator.h>" namespace "OpenMS":

  cdef cppclass PeakTypeEstimator:
```

```
    PeakTypeEstimator()

    int estimateType(libcpp_vector[Peak1D].iterator,
             libcpp_vector[Peak1D].iterator)  # wrap-ignore
```

6.302 PeakWidthEstimator

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/PeakWidthEstimator.h>"\
       namespace "OpenMS":

  cdef cppclass PeakWidthEstimator "OpenMS::PeakWidthEstimator":

    PeakWidthEstimator(PeakWidthEstimator)  #wrap-ignore
    PeakWidthEstimator(MSExperiment[Peak1D,ChromatogramPeak] exp_picked,
               libcpp_vector[libcpp_vector[PeakBoundary] ] & boundaries)

    double getPeakWidth(double mz)
```

6.303 PepIterator

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/PepIterator.h>" namespace "OpenMS":

  cdef cppclass PepIterator "OpenMS::PepIterator":
    # wrap-ignore
    # ABSTRACT class
    PepIterator()
    PepIterator(PepIterator)
    # POINTER # FASTAEntry operator*()
    # PepIterator  operator++()
    # POINTER # PepIterator * operator++(int )
    void setFastaFile(String & f)
    String getFastaFile()
    void setSpectrum(libcpp_vector[ double ] & s)
    libcpp_vector[ double ]  getSpectrum()
    void setTolerance(double t)
    double getTolerance()
    bool begin()
    bool isAtEnd()
    void registerChildren()
```

6.304 PepXMLFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/PepXMLFile.h>" namespace "OpenMS":

  cdef cppclass PepXMLFile:

    PepXMLFile()

    void load(String filename,
          libcpp_vector[ProteinIdentification] & protein_ids,
          libcpp_vector[PeptideIdentification] & peptide_ids
          )

    void load(String filename,
          libcpp_vector[ProteinIdentification] & protein_ids,
          libcpp_vector[PeptideIdentification] & peptide_ids,
          String experiment_name
          )

    void load(String filename,
          libcpp_vector[ProteinIdentification] & protein_ids,
          libcpp_vector[PeptideIdentification] & peptide_ids,
          String experiment_name,
          SpectrumMetaDataLookup lookup
          )

    void store(String filename,
          libcpp_vector[ProteinIdentification] & protein_ids,
          libcpp_vector[PeptideIdentification] & peptide_ids
          )

    void store(String filename,
          libcpp_vector[ProteinIdentification] & protein_ids,
          libcpp_vector[PeptideIdentification] & peptide_ids,
          String mz_file,
          String mz_name,
          bool peptideprophet_analyzed
          )

    void keepNativeSpectrumName(bool keep)
```

6.305 PepXMLFileMascot

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/PepXMLFileMascot.h>" namespace "OpenMS":

  cdef cppclass PepXMLFileMascot :

    PepXMLFileMascot()
    PepXMLFileMascot(PepXMLFileMascot)  #wrap-ignore
```

6.306 PeptideAndProteinQuant

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/QUANTITATION/PeptideAndProteinQuant.h>" namespace\
       "OpenMS":

  cdef cppclass PeptideAndProteinQuant(DefaultParamHandler):
    # wrap-inherits:
    #  DefaultParamHandler

    PeptideAndProteinQuant()
    PeptideAndProteinQuant(PeptideAndProteinQuant)  #wrap-ignore

    void readQuantData(FeatureMap & map_in)
    void readQuantData(ConsensusMap & map_in)
    void readQuantData(libcpp_vector[ProteinIdentification] & proteins,
                libcpp_vector[PeptideIdentification] & peptides)

    void quantifyPeptides(libcpp_vector[PeptideIdentification] & peptides)
    void quantifyProteins(ProteinIdentification & proteins)

    PeptideAndProteinQuant_Statistics getStatistics()

cdef extern from "<OpenMS/ANALYSIS/QUANTITATION/PeptideAndProteinQuant.h>" namespace\
       "OpenMS::PeptideAndProteinQuant":

  cdef cppclass PeptideAndProteinQuant_Statistics\
       "OpenMS::PeptideAndProteinQuant::Statistics":
    Size n_samples

    Size quant_proteins
    Size too_few_peptides
```

```
    Size quant_peptides
    Size total_peptides

    Size quant_features
    Size total_features
    Size blank_features
    Size ambig_features

    PeptideAndProteinQuant_Statistics()
    PeptideAndProteinQuant_Statistics(PeptideAndProteinQuant_Statistics) #\
       wrap-ignore

  cdef cppclass PeptideAndProteinQuant_PeptideData\
       "OpenMS::PeptideAndProteinQuant::PeptideData":

    libcpp_set[String] accessions

    Size id_count

    PeptideAndProteinQuant_PeptideData()
    PeptideAndProteinQuant_PeptideData(PeptideAndProteinQuant_PeptideData) #\
       wrap-ignore

  cdef cppclass PeptideAndProteinQuant_ProteinData\
       "OpenMS::PeptideAndProteinQuant::ProteinData":

    Size id_count

    PeptideAndProteinQuant_ProteinData()
    PeptideAndProteinQuant_ProteinData(PeptideAndProteinQuant_ProteinData) #\
       wrap-ignore
```

6.307 PeptideEvidence

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/PeptideEvidence.h>" namespace "OpenMS":
  cdef cppclass PeptideEvidence :
    PeptideEvidence()
    PeptideEvidence(PeptideEvidence)
    void setStart(Int start)
    Int getStart()
    void setEnd(Int end)
    Int getEnd()
```

```
    void setAABefore(char rhs)
    char getAABefore()
    void setAAAfter(char rhs)
    char getAAAfter()
    void setProteinAccession(String s)
    String getProteinAccession()
    bool operator==(PeptideEvidence & rhs)
    bool operator!=(PeptideEvidence & rhs)
```

6.308 PeptideHit

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/PeptideHit.h>" namespace "OpenMS":

  cdef cppclass PeptideHit:

    PeptideHit()

    PeptideHit(double score,
           UInt     rank,
           Int    charge,
           AASequence sequence
           )

    PeptideHit(PeptideHit)  # wrap-ignore

    float getScore()
    UInt getRank()
    AASequence getSequence()
    Int getCharge()
    libcpp_vector[PeptideEvidence] getPeptideEvidences()
    void setPeptideEvidences(libcpp_vector[PeptideEvidence])
    void addPeptideEvidence(PeptideEvidence)
    libcpp_set[String] extractProteinAccessions()

    void setAnalysisResults(libcpp_vector[PeptideHit_AnalysisResult] aresult)
    void addAnalysisResults(PeptideHit_AnalysisResult aresult)
    libcpp_vector[PeptideHit_AnalysisResult] getAnalysisResults()

    void setScore(double)
    void setRank(UInt)
    void setSequence(AASequence)
    void setCharge(Int)
```

```
    bool operator==(PeptideHit)
    bool operator!=(PeptideHit)
    bool isMetaEmpty()
    void clearMetaInfo()

    void getKeys(libcpp_vector[String] & keys)
    void getKeys(libcpp_vector[unsigned int] & keys)  # wrap-as:getKeysAsIntegers
    DataValue getMetaValue(unsigned int)
    DataValue getMetaValue(String)
    void setMetaValue(unsigned int, DataValue)
    void setMetaValue(String, DataValue)
    bool metaValueExists(String)
    bool metaValueExists(unsigned int)
    void removeMetaValue(String)
    void removeMetaValue(unsigned int)

  cdef cppclass PeptideHit_AnalysisResult\
       "OpenMS::PeptideHit::PepXMLAnalysisResult":

    PeptideHit_AnalysisResult()
    PeptideHit_AnalysisResult(PeptideHit_AnalysisResult)  #wrap-ignore

    String score_type
    bool higher_is_better
    double main_score
    libcpp_map[String, double] sub_scores
```

6.309 PeptideIdentification

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/PeptideIdentification.h>" namespace "OpenMS":

  cdef cppclass PeptideIdentification(MetaInfoInterface):
    # wrap-inherits:
    #  MetaInfoInterface

    PeptideIdentification()
    PeptideIdentification(PeptideIdentification)
    bool operator==(PeptideIdentification)
    bool operator!=(PeptideIdentification)

    libcpp_vector[PeptideHit] getHits()
    void insertHit(PeptideHit)
    void setHits(libcpp_vector[PeptideHit])
```

```
double getSignificanceThreshold()
void setSignificanceThreshold(double value)

String   getScoreType()
void     setScoreType(String)
bool     isHigherScoreBetter()
void     setHigherScoreBetter(bool)
String   getIdentifier()
void     setIdentifier(String)

bool     hasMZ()
double   getMZ()
void     setMZ(double)

bool     hasRT()
double   getRT()
void     setRT(double)

String   getBaseName()
void     setBaseName(String)

String   getExperimentLabel()
void     setExperimentLabel(String)

void     assignRanks()
void     sort()
void sortByRank()
bool     empty()

libcpp_vector[PeptideHit] getReferencingHits(libcpp_vector[PeptideHit],\
   libcpp_set[String] &)

void getKeys(libcpp_vector[String] & keys)
void getKeys(libcpp_vector[unsigned int] & keys)  # wrap-as:getKeysAsIntegers
DataValue getMetaValue(unsigned int)
DataValue getMetaValue(String)
void setMetaValue(unsigned int, DataValue)
void setMetaValue(String, DataValue)
bool metaValueExists(String)
bool metaValueExists(unsigned int)
void removeMetaValue(String)
void removeMetaValue(unsigned int)
```

6.310 PeptideIndexing

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/ID/PeptideIndexing.h>" namespace "OpenMS":

  cdef cppclass PeptideIndexing(DefaultParamHandler) :
    # wrap-inherits:
    #  DefaultParamHandler
    PeptideIndexing()
    PeptideIndexing(PeptideIndexing)  #wrap-ignore

    PeptideIndexing_ExitCodes run(libcpp_vector[ FASTAEntry ] & proteins,
                   libcpp_vector[ ProteinIdentification ] & prot_ids,
                   libcpp_vector[ PeptideIdentification ] & pep_ids)

cdef extern from "<OpenMS/ANALYSIS/ID/PeptideIndexing.h>" namespace\
       "OpenMS::PeptideIndexing":
  cdef enum PeptideIndexing_ExitCodes "OpenMS::PeptideIndexing::ExitCodes":
    #wrap-attach:
    #  PeptideIndexing
    EXECUTION_OK
    DATABASE_EMPTY
    PEPTIDE_IDS_EMPTY
    DATABASE_CONTAINS_MULTIPLES
    ILLEGAL_PARAMETERS
    UNEXPECTED_RESULT
```

6.311 PeptideProteinResolution

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/ID/PeptideProteinResolution.h>" namespace\
       "OpenMS":

  cdef cppclass PeptideProteinResolution "OpenMS::PeptideProteinResolution":
    PeptideProteinResolution(PeptideProteinResolution)  #wrap-ignore
    PeptideProteinResolution(bool statistics)

    void buildGraph(ProteinIdentification & protein,
            libcpp_vector[ PeptideIdentification ] & peptides)

    void resolveGraph(ProteinIdentification & protein,
              libcpp_vector[ PeptideIdentification ] & peptides)

    PeptideProteinResolution_ConnectedComponent findConnectedComponent(Size &\
       root_prot_grp)
```

```
    void resolveConnectedComponent(PeptideProteinResolution_ConnectedComponent &\
       conn_comp,
                     ProteinIdentification & protein,
                     libcpp_vector[ PeptideIdentification ] &
                     peptides)

cdef extern from "<OpenMS/ANALYSIS/ID/PeptideProteinResolution.h>" namespace\
       "OpenMS":

  cdef cppclass PeptideProteinResolution_ConnectedComponent\
       "OpenMS::ConnectedComponent":
    PeptideProteinResolution_ConnectedComponent()
   \
       PeptideProteinResolution_ConnectedComponent(PeptideProteinResolution_ConnectedCompon\
      ent)  #wrap-ignore
    libcpp_set[ size_t ] prot_grp_indices
    libcpp_set[ size_t ] pep_indices
```

6.312 PercolatorOutfile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/PercolatorOutfile.h>" namespace "OpenMS":

  cdef cppclass PercolatorOutfile "OpenMS::PercolatorOutfile":
    PercolatorOutfile()
    PercolatorOutfile(PercolatorOutfile)  #wrap-ignore
    # libcpp_string score_type_names()
    PercolatorOutfile_ScoreType getScoreType(String score_type_name)
    void load(String & filename, ProteinIdentification & proteins,
          libcpp_vector[ PeptideIdentification ] & peptides,
          SpectrumMetaDataLookup & lookup,
          PercolatorOutfile_ScoreType output_score)

cdef extern from "<OpenMS/FORMAT/PercolatorOutfile.h>" namespace\
       "OpenMS::PercolatorOutfile":
  cdef enum PercolatorOutfile_ScoreType "OpenMS::PercolatorOutfile::ScoreType":
    #wrap-attach:
    #  PercolatorOutfile
    QVALUE
    POSTERRPROB
    SCORE
    SIZE_OF_SCORETYPE
```

6.313 PosteriorErrorProbabilityModel

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/MATH/STATISTICS/PosteriorErrorProbabilityModel.h>"\
       namespace "OpenMS::Math":

  cdef cppclass PosteriorErrorProbabilityModel(DefaultParamHandler):
    # wrap-inherits:
    #  DefaultParamHandler

    PosteriorErrorProbabilityModel()
    PosteriorErrorProbabilityModel(PosteriorErrorProbabilityModel)    # wrap-ignore

    bool fit(libcpp_vector[double] & search_engine_scores)
    bool fit(libcpp_vector[double] & search_engine_scores, libcpp_vector[double] &\
       probabilities)

    void fillDensities(libcpp_vector[double] & x_scores, libcpp_vector[double] &\
       incorrect_density, libcpp_vector[double] & correct_density)
    double computeMaxLikelihood(libcpp_vector[double] & incorrect_density,\
       libcpp_vector[double] & correct_density)

    double one_minus_sum_post(libcpp_vector[double] & incorrect_density,\
       libcpp_vector[double] & correct_density)
    double sum_post(libcpp_vector[double] & incorrect_density, libcpp_vector[double]\
       & correct_density)
    double sum_pos_x0(libcpp_vector[double] & x_scores, libcpp_vector[double] &\
       incorrect_density, libcpp_vector[double] & correct_density)
    double sum_neg_x0(libcpp_vector[double] & x_scores, libcpp_vector[double] &\
       incorrect_density, libcpp_vector[double] & correct_density)
    double sum_pos_sigma(libcpp_vector[double] & x_scores, libcpp_vector[double] &\
       incorrect_density, libcpp_vector[double] & correct_density, double positive_mean)
    double sum_neg_sigma(libcpp_vector[double] & x_scores, libcpp_vector[double] &\
       incorrect_density, libcpp_vector[double] & correct_density, double positive_mean)

    GaussFitResult getCorrectlyAssignedFitResult()

    GaussFitResult getIncorrectlyAssignedFitResult()

    double getNegativePrior()

    double getGauss(double x, GaussFitResult & params)

    double getGumbel(double x, GaussFitResult & params)

    double computeProbability(double score)
```

```
    TextFile initPlots(libcpp_vector[ double ] & x_scores)

    String getGumbelGnuplotFormula(GaussFitResult & params)

    String getGaussGnuplotFormula(GaussFitResult & params)

    String getBothGnuplotFormula(GaussFitResult & incorrect, GaussFitResult &\
       correct)

    void plotTargetDecoyEstimation(libcpp_vector[double] & target,\
       libcpp_vector[double] & decoy)

    double getSmallestScore()

    void tryGnuplot(String & gp_file)
```

6.314 Precursor

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/Precursor.h>" namespace "OpenMS":

  cdef cppclass Precursor(Peak1D):

    # wrap-inherits:
    #  Peak1D

    void setCVTerms(libcpp_vector[CVTerm] & terms)
    void replaceCVTerm(CVTerm & term)

    void replaceCVTerms(libcpp_vector[CVTerm] cv_terms,
                String accession
               )

    void replaceCVTerms(Map[String, libcpp_vector[CVTerm] ] cv_term_map
               )

    Map[String, libcpp_vector[CVTerm] ] getCVTerms()
    void addCVTerm(CVTerm & term)

    bool hasCVTerm(String accession)
    bool empty()

    void getKeys(libcpp_vector[String] & keys)
```

```
    void getKeys(libcpp_vector[unsigned int] & keys)  # wrap-as:getKeysAsIntegers
    DataValue getMetaValue(unsigned int)
    DataValue getMetaValue(String)
    void setMetaValue(unsigned int, DataValue)
    void setMetaValue(String, DataValue)
    bool metaValueExists(String)
    bool metaValueExists(unsigned int)
    void removeMetaValue(String)
    void removeMetaValue(unsigned int)

    Precursor()
    Precursor(Precursor)

    libcpp_set[ActivationMethod] getActivationMethods()
    void setActivationMethods(libcpp_set[ActivationMethod] activation_methods)

    double getActivationEnergy()
    void setActivationEnergy(double activation_energy)

    double getIsolationWindowLowerOffset()
    void setIsolationWindowLowerOffset(double bound)

    double getIsolationWindowUpperOffset()
    void setIsolationWindowUpperOffset(double bound)

    int getCharge()
    void setCharge(int charge)

    libcpp_vector[int] getPossibleChargeStates()
    void setPossibleChargeStates(libcpp_vector[int] possible_charge_states)

    double getUnchargedMass()

    bool operator==(Precursor)
    bool operator!=(Precursor)

cdef extern from "<OpenMS/METADATA/Precursor.h>" namespace "OpenMS::Precursor":
  cdef enum ActivationMethod:
    CID,            #< Collision-induced dissociation
    PSD,            #< Post-source decay
    PD,             #< Plasma desorption
    SID,            #< Surface-induced dissociation
    BIRD,               #< Blackbody infrared radiative dissociation
    ECD,                #< Electron capture dissociation
    IMD,                #< Infrared multiphoton dissociation
    SORI,               #< Sustained off-resonance irradiation
    HCID,               #< High-energy collision-induced dissociation
    LCID,               #< Low-energy collision-induced dissociation
```

```
    PHD,                #< Photodissociation
    ETD,                #< Electron transfer dissociation
    PQD,                #< Pulsed q dissociation
    SIZE_OF_ACTIVATIONMETHOD
```

6.315 PrecursorIonSelection

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/TARGETED/PrecursorIonSelection.h>" namespace\
       "OpenMS":

  cdef cppclass PrecursorIonSelection(DefaultParamHandler) :
    # wrap-inherits:
    #  DefaultParamHandler
    PrecursorIonSelection()
    PrecursorIonSelection(PrecursorIonSelection)
    double  getMaxScore()
    void setMaxScore(double & max_score)
    void sortByTotalScore(FeatureMap & features)
    void getNextPrecursors(FeatureMap & features, FeatureMap & next_features, UInt\
       number)

    void getNextPrecursors(libcpp_vector[ int ] & solution_indices,
                  libcpp_vector[ IndexTriple ] & variable_indices,
                  libcpp_set[ int ] & measured_variables,
                  FeatureMap & features,
                  FeatureMap & new_features,
                  UInt step_size,
                  PSLPFormulation & ilp)
    void rescore(FeatureMap & features, libcpp_vector[ PeptideIdentification ] &\
       new_pep_ids, libcpp_vector[ ProteinIdentification ] & prot_ids,\
       PrecursorIonSelectionPreprocessing & preprocessed_db, bool check_meta_values)
    void simulateRun(FeatureMap & features, libcpp_vector[ PeptideIdentification ] &\
       pep_ids, libcpp_vector[ ProteinIdentification ] & prot_ids,\
       PrecursorIonSelectionPreprocessing & preprocessed_db, String path,\
       MSExperiment[Peak1D, ChromatogramPeak] & experiment, String precursor_path)
    void setLPSolver(SOLVER solver)
    SOLVER getLPSolver()
    void reset()
cdef extern from "<OpenMS/ANALYSIS/TARGETED/PrecursorIonSelection.h>" namespace\
       "OpenMS::PrecursorIonSelection":
  cdef enum PrecursorIonSelection_Type "OpenMS::PrecursorIonSelection::Type":
    #wrap-attach:
    #  PrecursorIonSelection
```

```
    IPS
    ILP_IPS
    SPS
    UPSHIFT
    DOWNSHIFT
    DEX

#     SeqTotalScoreMore(SeqTotalScoreMore)  #wrap-ignore
#     bool operator()(Feature & left, Feature & right)  # wrap-cast:evaluate
#
# cdef extern from "<OpenMS/ANALYSIS/TARGETED/PrecursorIonSelection.h>" namespace\
       "OpenMS::PrecursorIonSelection":
#
#   cdef cppclass TotalScoreMore :
#     TotalScoreMore(TotalScoreMore)  #wrap-ignore
#     bool operator()(Feature & left, Feature & right)  # wrap-cast:evaluate
#
```

6.316 PrecursorIonSelectionPreprocessing

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/TARGETED/PrecursorIonSelectionPreprocessing.h>"\
       namespace "OpenMS":

  cdef cppclass PrecursorIonSelectionPreprocessing(DefaultParamHandler) :
    # wrap-inherits:
    #  DefaultParamHandler
    PrecursorIonSelectionPreprocessing()
    PrecursorIonSelectionPreprocessing(PrecursorIonSelectionPreprocessing)
    # TODO STL map
    # libcpp_map[ String, libcpp_vector[ double ] ]  getProtMasses()
    libcpp_vector[ double ]  getMasses(String acc)
    # libcpp_map[ String, libcpp_vector[ double ] ]  getProteinRTMap()
    # libcpp_map[ String, libcpp_vector[ double ] ]  getProteinPTMap()
    # libcpp_map[ String, libcpp_vector[ String ] ]  getProteinPeptideSequenceMap()
    void dbPreprocessing(String db_path, bool save)
    void dbPreprocessing(String db_path, String rt_model_path, String dt_model_path,\
       bool save)
    void loadPreprocessing()
    double getWeight(double mass)
    double getRT(String prot_id, Size peptide_index)
    double getPT(String prot_id, Size peptide_index)
    void setFixedModifications(StringList & modifications)
    # libcpp_map[ char, libcpp_vector[ String ] ]  getFixedModifications()
    void setGaussianParameters(double mu, double sigma)
```

```
    double getGaussMu()
    double getGaussSigma()
    double getRTProbability(String prot_id, Size peptide_index, Feature & feature)
    double getRTProbability(double pred_rt, Feature & feature)
```

6.317 Product

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/Product.h>" namespace "OpenMS":

  cdef cppclass Product:

    Product()
    Product(Product)

    bool operator==(Product)
    bool operator!=(Product)

    double getMZ()
    void setMZ(double)

    double getIsolationWindowLowerOffset()
    void  setIsolationWindowLowerOffset(double bound)

    double getIsolationWindowUpperOffset()
    void  setIsolationWindowUpperOffset(double bound)
```

6.318 ProgressLogger

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CONCEPT/ProgressLogger.h>" namespace "OpenMS":

  cdef cppclass ProgressLogger:
    ProgressLogger()
    void setLogType(LogType)
    LogType getLogType()
    void startProgress(SignedSize begin, SignedSize end, String label)
    void setProgress(SignedSize value)
    void endProgress()

cdef extern from "<OpenMS/CONCEPT/ProgressLogger.h>" namespace\
```

```
      "OpenMS::ProgressLogger":

  cdef enum LogType:
    # wrap-attach: ProgressLogger
    CMD, GUI, NONE
```

6.319 ProtXMLFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/ProtXMLFile.h>" namespace "OpenMS":

  cdef cppclass ProtXMLFile:

    ProtXMLFile()
    ProtXMLFile(ProtXMLFile)  # wrap-ignore

    void load(String filename, ProteinIdentification & protein_ids,\
       PeptideIdentification & peptide_ids)

    void store(String filename, ProteinIdentification & protein_ids,\
       PeptideIdentification & peptide_ids, String document_id)
```

6.320 ProteinHit

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/ProteinHit.h>" namespace "OpenMS":

  cdef cppclass ProteinHit:

    ProteinHit()
    ProteinHit(double score, UInt rank, String accession, String sequence)
    ProteinHit(ProteinHit)

    float getScore()
    UInt getRank()
    String getSequence()
    String getAccession()
    String getDescription()
    double getCoverage()
```

```
    void setScore(float )
    void setRank(UInt)
    void setSequence(String)
    void setAccession(String)
    void setDescription(String description)
    void setCoverage(double)

    bool operator==(ProteinHit)
    bool operator!=(ProteinHit)
    bool isMetaEmpty()
    void clearMetaInfo()

    void getKeys(libcpp_vector[String] & keys)
    void getKeys(libcpp_vector[unsigned int] & keys)  # wrap-as:getKeysAsIntegers
    DataValue getMetaValue(unsigned int)
    DataValue getMetaValue(String)
    void setMetaValue(unsigned int, DataValue)
    void setMetaValue(String, DataValue)
    bool metaValueExists(String)
    bool metaValueExists(unsigned int)
    void removeMetaValue(String)
    void removeMetaValue(unsigned int)
```

6.321 ProteinIdentification

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/ProteinIdentification.h>" namespace "OpenMS":

  cdef cppclass ProteinIdentification(MetaInfoInterface):
    # wrap-inherits:
    #  MetaInfoInterface

    ProteinIdentification()
    ProteinIdentification(ProteinIdentification)

    bool operator==(ProteinIdentification)
    bool operator!=(ProteinIdentification)

    void getKeys(libcpp_vector[String] & keys)
    void getKeys(libcpp_vector[unsigned int] & keys)  # wrap-as:getKeysAsIntegers
    DataValue getMetaValue(unsigned int)
    DataValue getMetaValue(String)
    void setMetaValue(unsigned int, DataValue)
    void setMetaValue(String, DataValue)
```

```
    bool metaValueExists(String)
    bool metaValueExists(unsigned int)
    void removeMetaValue(String)
    void removeMetaValue(unsigned int)

    libcpp_vector[ProteinHit] getHits()
    void insertHit(ProteinHit input)
    void setHits(libcpp_vector[ProteinHit] hits)
    libcpp_vector[ProteinGroup] getProteinGroups()
    void insertProteinGroup(ProteinGroup group)

    libcpp_vector[ProteinGroup] getIndistinguishableProteins()
    void insertIndistinguishableProteins(ProteinGroup group)

    double getSignificanceThreshold()
    void setSignificanceThreshold(double value)
    String getScoreType()
    void setScoreType(String type)
    bool isHigherScoreBetter()
    void setHigherScoreBetter(bool higher_is_better)
    void sort()
    void assignRanks()

    void computeCoverage(libcpp_vector[PeptideIdentification] pep_ids)

    DateTime getDateTime()
    void setDateTime(DateTime date)
    void setSearchEngine(String search_engine)
    String getSearchEngine()
    void setSearchEngineVersion(String search_engine_version)
    String getSearchEngineVersion()
    void setSearchParameters(SearchParameters search_parameters)
    SearchParameters getSearchParameters()
    String getIdentifier()
    void setIdentifier(String id_)

    void setPrimaryMSRunPath(StringList& s)
    StringList getPrimaryMSRunPath()

cdef extern from "<OpenMS/METADATA/ProteinIdentification.h>" namespace\
       "OpenMS::ProteinIdentification":

  cdef enum PeakMassType:
    # wrap-attach:
    #  ProteinIdentification
    MONOISOTOPIC, AVERAGE, SIZE_OF_PEAKMASSTYPE

  cdef cppclass ProteinGroup:
```

```
    ProteinGroup()
    ProteinGroup(ProteinGroup)

    double probability
    StringList accessions

  cdef cppclass SearchParameters(MetaInfoInterface):
    # wrap-inherits:
    #  MetaInfoInterface

    SearchParameters()
    SearchParameters(SearchParameters)

    String db      #< The used database
    String db_version      #< The database version
    String taxonomy      #< The taxonomy restriction
    String charges      #< The allowed charges for the search
    PeakMassType mass_type      #< Mass type of the peaks
    libcpp_vector[String] fixed_modifications      #< Used fixed modifications
    libcpp_vector[String] variable_modifications      #< Allowed variable\
       modifications
    UInt missed_cleavages      #< The number of allowed missed cleavages
    double fragment_mass_tolerance      #< Mass tolerance of fragment ions (Dalton)
    bool fragment_mass_tolerance_ppm
    double precursor_mass_tolerance      #< Mass tolerance of precursor ions\
       (Dalton)
    bool precursor_mass_tolerance_ppm
    Enzyme digestion_enzyme      #< The enzyme for cleavage
```

6.322 ProteinInference

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/QUANTITATION/ProteinInference.h>" namespace\
       "OpenMS":

  cdef cppclass ProteinInference "OpenMS::ProteinInference":
    ProteinInference()
    ProteinInference(ProteinInference)
    void infer(ConsensusMap & consensus_map, UInt reference_map)
```

6.323 ProteinResolver

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/QUANTITATION/ProteinResolver.h>" namespace\
       "OpenMS":
  cdef cppclass ProteinResolver(DefaultParamHandler):
    # wrap-inherits:
    #  DefaultParamHandler

    ProteinResolver()
    ProteinResolver(ProteinResolver)

    void resolveConsensus(ConsensusMap & consensus)
    void resolveID(libcpp_vector[PeptideIdentification] & peptide_identifications)
    void setProteinData(libcpp_vector[FASTAEntry] & protein_data)
    libcpp_vector[ResolverResult] getResults()

    void countTargetDecoy(libcpp_vector[ MSDGroup ] & msd_groups, ConsensusMap &\
       consensus)
    void countTargetDecoy(libcpp_vector[ MSDGroup ] & msd_groups, libcpp_vector[\
       PeptideIdentification ] & peptide_nodes)
    void clearResult()

    PeptideIdentification getPeptideIdentification(ConsensusMap & consensus,\
       PeptideEntry * peptide)
    PeptideHit getPeptideHit(ConsensusMap & consensus, PeptideEntry * peptide)
    PeptideIdentification getPeptideIdentification(libcpp_vector[\
       PeptideIdentification ] & peptide_nodes, PeptideEntry * peptide)
    PeptideHit getPeptideHit(libcpp_vector[ PeptideIdentification ] & peptide_nodes,\
       PeptideEntry * peptide)

cdef extern from "<OpenMS/ANALYSIS/QUANTITATION/ProteinResolver.h>" namespace\
       "OpenMS::ProteinResolver":

  cdef cppclass ISDGroup "OpenMS::ProteinResolver::ISDGroup":
    ISDGroup()
    ISDGroup(ISDGroup)  #wrap-ignore
    # NAMESPACE # # POINTER # std::list[ ProteinEntry * ] proteins
    # NAMESPACE # # POINTER # std::list[ PeptideEntry * ] peptides
    Size index
    # libcpp_list[ size_t ] msd_groups # TODO no converter for List

cdef extern from "<OpenMS/ANALYSIS/QUANTITATION/ProteinResolver.h>" namespace\
       "OpenMS::ProteinResolver":

  cdef cppclass MSDGroup "OpenMS::ProteinResolver::MSDGroup":
```

```
    MSDGroup()
    MSDGroup(MSDGroup)  #wrap-ignore
    # NAMESPACE # # POINTER # std::list[ ProteinEntry * ] proteins
    # NAMESPACE # # POINTER # std::list[ PeptideEntry * ] peptides
    Size index
    # POINTER # ISDGroup * isd_group
    Size number_of_decoy
    Size number_of_target
    Size number_of_target_plus_decoy
    float intensity

cdef extern from "<OpenMS/ANALYSIS/QUANTITATION/ProteinResolver.h>" namespace\
       "OpenMS::ProteinResolver":

  cdef cppclass ResolverResult "OpenMS::ProteinResolver::ResolverResult":
    ResolverResult()
    ResolverResult(ResolverResult)

    String identifier
cdef extern from "<OpenMS/ANALYSIS/QUANTITATION/ProteinResolver.h>" namespace\
       "OpenMS::ProteinResolver":

  cdef cppclass PeptideEntry "OpenMS::ProteinResolver::PeptideEntry":
    PeptideEntry()
    PeptideEntry(PeptideEntry)  #wrap-ignore
    # NAMESPACE # # POINTER # std::list[ ProteinEntry * ] proteins
    bool traversed
    String sequence
    Size peptide_identification
    Size peptide_hit
    Size index
    Size msd_group
    Size isd_group
    bool experimental
    float intensity
    String origin

cdef extern from "<OpenMS/ANALYSIS/QUANTITATION/ProteinResolver.h>" namespace\
       "OpenMS::ProteinResolver::ResolverResult":

  cdef enum ProteinResolverResult_Type\
       "OpenMS::ProteinResolver::ResolverResult::type":
    PeptideIdent
    Consensus

cdef extern from "<OpenMS/ANALYSIS/QUANTITATION/ProteinResolver.h>" namespace\
       "OpenMS::ProteinResolver":
```

```
  cdef cppclass ProteinEntry "OpenMS::ProteinResolver::ProteinEntry":
    ProteinEntry()
    ProteinEntry(ProteinEntry)  #wrap-ignore
    # NAMESPACE # # POINTER # std::list[ PeptideEntry * ] peptides
    bool traversed
    # NAMESPACE # # POINTER # FASTAFile::FASTAEntry * fasta_entry
    ProteinEntry_type protein_type
    double weight
    float coverage
    # NAMESPACE # # POINTER # std::list[ ProteinEntry * ] indis
    Size index
    Size msd_group
    Size isd_group
    Size number_of_experimental_peptides

cdef extern from "<OpenMS/ANALYSIS/QUANTITATION/ProteinResolver.h>" namespace\
       "OpenMS::ProteinResolver::ProteinEntry":
  cdef enum ProteinEntry_type "OpenMS::ProteinResolver::ProteinEntry::type":
    #wrap-attach:
    #  ProteinEntry
    primary
    secondary
    primary_indistinguishable
    secondary_indistinguishable
```

6.324 ProtonDistributionModel

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/ID/ProtonDistributionModel.h>" namespace\
       "OpenMS":

  cdef cppclass ProtonDistributionModel(DefaultParamHandler) :
    # wrap-inherits:
    #  DefaultParamHandler
    ProtonDistributionModel()
    ProtonDistributionModel(ProtonDistributionModel)
    void getProtonDistribution(libcpp_vector[ double ] & bb_charges, libcpp_vector[\
       double ] & sc_charges, AASequence & peptide, Int charge, ResidueType res_type)
    void getChargeStateIntensities(AASequence & peptide, AASequence & n_term_ion,\
       AASequence & c_term_ion, Int charge, ResidueType n_term_type, libcpp_vector[ double\
       ] & n_term_intensities, libcpp_vector[ double ] & c_term_intensities,\
       FragmentationType type_)
    void setPeptideProtonDistribution(libcpp_vector[ double ] & bb_charge,\
       libcpp_vector[ double ] & sc_charge)
```

```
cdef extern from "<OpenMS/ANALYSIS/ID/ProtonDistributionModel.h>" namespace\
       "OpenMS::ProtonDistributionModel":
  cdef enum FragmentationType "OpenMS::ProtonDistributionModel::FragmentationType":
    #wrap-attach:
    #  ProtonDistributionModel
    ChargeDirected
    ChargeRemote
    SideChain
```

6.325 PythonMSDataConsumer

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "python_ms_data_consumer.hpp":

  cdef cppclass PythonMSDataConsumer(IMSDataConsumer[Peak1D, ChromatogramPeak]):
    # wrap-ignore

    PythonMSDataConsumer(object py_consumer,
                object (*spectrum_wrapper)(const MSSpectrum[Peak1D] &),
                object (*chromatogram_wrapper)(const MSChromatogram[ChromatogramPeak] &),
                object (*experimental_settings_wrapper)(const ExperimentalSettings &)
               )

    void consumeSpectrum(MSSpectrum[Peak1D] &) except +

    void consumeChromatogram(MSChromatogram[ChromatogramPeak] &) except +

    void setExpectedSize(Size ns, Size, nc) except +

    void setExperimentalSettings(ExperimentalSettings &) except +
```

6.326 QTCluster

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/DATASTRUCTURES/QTCluster.h>" namespace "OpenMS":

  cdef cppclass QTCluster "OpenMS::QTCluster":
    QTCluster(QTCluster)  #wrap-ignore
    # POINTER #  QTCluster(GridFeature * center_point, Size num_maps, double\
```

```
      max_distance, bool use_IDs)
    double getCenterRT()
    double getCenterMZ()
    Int getXCoord()
    Int getYCoord()
    Size size()
    bool operator<(QTCluster & cluster)
    # POINTER # void add(GridFeature * element, double distance)
    # POINTER # void getElements(libcpp_map[ Size, GridFeature * ] & elements)
    # POINTER # bool update(libcpp_map[ Size, GridFeature * ] & removed)
    double getQuality()
    libcpp_set[ AASequence ]  getAnnotations()
    void setInvalid()
    bool isInvalid()
    void initializeCluster()
    void finalizeCluster()
    # NAMESPACE # # POINTER # OpenMSBoost::unordered_map[ Size, libcpp_vector[\
      GridFeature * ] ] getAllNeighbors()
    # NeighborMap getNeighbors()
```

6.327 QTClusterFinder

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/MAPMATCHING/QTClusterFinder.h>" namespace\
       "OpenMS":

  cdef cppclass QTClusterFinder(BaseGroupFinder) :
    # wrap-inherits:
    #  BaseGroupFinder
    QTClusterFinder()
    QTClusterFinder(QTClusterFinder)  #wrap-ignore
    void run(libcpp_vector[ ConsensusMap ] & input_maps, ConsensusMap & result_map)
    void run(libcpp_vector[ FeatureMap ] & input_maps, ConsensusMap & result_map)
    String getProductName()
    # POINTER # BaseGroupFinder * create()
```

6.328 QcMLFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/QcMLFile.h>" namespace "OpenMS":
```

```
  cdef cppclass QcMLFile(XMLHandler,XMLFile,ProgressLogger) :
    # wrap-inherits:
    #  XMLHandler
    #  XMLFile
    #  ProgressLogger
    QcMLFile()
    QcMLFile(QcMLFile)  #wrap-ignore
    String map2csv(libcpp_map[ String, libcpp_map[ String, String ] ] & cvs_table,\
       String & separator)  # wrap-ignore
    String exportIDstats(String & filename)
    void addRunQualityParameter(String r, QualityParameter qp)
    void addRunAttachment(String r, Attachment at)
    void addSetQualityParameter(String r, QualityParameter qp)
    void addSetAttachment(String r, Attachment at)
    void removeAttachment(String r, libcpp_vector[ String ] & ids, String at)
    void removeAttachment(String r, String at)
    void removeAllAttachments(String at)
    void removeQualityParameter(String r, libcpp_vector[ String ] & ids)
    void merge(QcMLFile & addendum, String setname)
    void collectSetParameter(String setname, String qp, libcpp_vector[ String ] & ret)
    String exportAttachment(String filename, String qpname)
    void getRunNames(libcpp_vector[ String ] & ids)
    bool existsRun(String filename)
    bool existsSet(String filename)
    void existsRunQualityParameter(String filename, String qpname, libcpp_vector[\
       String ] & ids)
    void existsSetQualityParameter(String filename, String qpname, libcpp_vector[\
       String ] & ids)
    void store(String & filename)
    void load(String & filename)

    void registerRun(String id_, String name)
    void registerSet(String id_, String name, libcpp_set[ String ] & names)
    String exportQP(String filename, String qpname)
    String exportQPs(String filename, StringList qpnames)
    void getRunIDs(libcpp_vector[ String ] & ids)

cdef extern from "<OpenMS/FORMAT/QcMLFile.h>" namespace "OpenMS::QcMLFile":

  cdef cppclass QualityParameter "OpenMS::QcMLFile::QualityParameter":
    QualityParameter()
    QualityParameter(QualityParameter)
    String name
    String id
    String value
    String cvRef
    String cvAcc
```

```
    String unitRef
    String unitAcc
    String flag
    bool operator==(QualityParameter & rhs)
    bool operator<(QualityParameter & rhs)
    bool operator>(QualityParameter & rhs)
    String toXMLString(UInt indentation_level)
```

6.329 QuantitativeExperimentalDesign

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/QUANTITATION/QuantitativeExperimentalDesign.h>"\
       namespace "OpenMS":

  cdef cppclass QuantitativeExperimentalDesign(DefaultParamHandler) :
    # wrap-inherits:
    #  DefaultParamHandler
    QuantitativeExperimentalDesign()
    QuantitativeExperimentalDesign(QuantitativeExperimentalDesign)  #wrap-ignore
    void applyDesign2Resolver(ProteinResolver & resolver, TextFile & file_,\
       StringList & fileNames)
    void applyDesign2Quantifier(PeptideAndProteinQuant & quantifier, TextFile &\
       file_, StringList & fileNames)
```

6.330 RANSAC

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/MATH/MISC/RANSAC.h>" namespace "OpenMS::Math":

  cdef cppclass RANSAC[TModelType]:
    # wrap-instances:
    #   RANSAC := RANSAC[RansacModelLinear]
    #   RANSACQuadratic := RANSAC[RansacModelQuadratic]

    RANSAC()
    RANSAC(RANSAC[TModelType] &)  # wrap-ignore

  cdef cppclass RANSACParam:

    RANSACParam()
```

```
    RANSACParam(RANSACParam)  # wrap-ignore
    RANSACParam(size_t p_n, size_t p_k, double p_t, size_t p_d, bool p_relative_d)

    libcpp_string toString()

    size_t n #; //< data points; The minimum number of data points required to fit\
       the model
    size_t k # ; //< iterations; The maximum number of iterations allowed in the\
       algorithm
    double t # ; //< Threshold value; for determining when a data point fits a\
       model. Corresponds to the maximal squared deviation in units of the _second_\
       dimension (dim2).
    size_t d # ; //< The number of close data values (according to 't') required to\
       assert that a model fits well to data
    bool relative_d #; //< Should 'd' be interpreted as percentages (0-100) of data\
       input size.
cdef extern from "<OpenMS/MATH/MISC/RANSAC.h>" namespace\
       "OpenMS::Math::RANSAC<OpenMS::Math::RansacModelLinear>":

  libcpp_vector[libcpp_pair[double,double]] ransac(
      libcpp_vector[libcpp_pair[double,double]] pairs,
      size_t n, size_t k, double t, size_t d, bool test
      )  # wrap-attach:RANSAC

cdef extern from "<OpenMS/MATH/MISC/RANSAC.h>" namespace\
       "OpenMS::Math::RANSAC<OpenMS:Math::RansacModelQuadratic>":

  libcpp_vector[libcpp_pair[double,double]] ransac(
      libcpp_vector[libcpp_pair[double,double]] pairs,
      size_t n, size_t k, double t, size_t d, bool test
      )  # wrap-attach:RANSACQuadratic
```

6.331 RANSACModelLinear

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/MATH/MISC/RANSACModelLinear.h>" namespace "OpenMS::Math":

  cdef cppclass RansacModelLinear:
     RansacModelLinear()
     RansacModelLinear(RansacModelLinear &)  # wrap-ignore
```

6.332 RANSACModelQuadratic

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/MATH/MISC/RANSACModelQuadratic.h>" namespace\
       "OpenMS::Math":

  cdef cppclass RansacModelQuadratic:
     RansacModelQuadratic()
     RansacModelQuadratic(RansacModelQuadratic &)  # wrap-ignore
```

6.333 RNPxlModificationsGenerator

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/RNPXL/RNPxlModificationsGenerator.h>" namespace\
       "OpenMS":

  cdef cppclass RNPxlModificationsGenerator:

    RNPxlModificationsGenerator()
    RNPxlModificationsGenerator(RNPxlModificationsGenerator)

    RNPxlModificationMassesResult initModificationMassesRNA(
      StringList target_nucleotides, StringList mappings,
      StringList restrictions, StringList modifications,
      String sequence_restriction, bool cysteine_adduct, Int max_length)

cdef extern from "<OpenMS/ANALYSIS/RNPXL/RNPxlModificationsGenerator.h>" namespace\
       "OpenMS":

  cdef cppclass RNPxlModificationMassesResult:

    RNPxlModificationMassesResult()
    RNPxlModificationMassesResult(RNPxlModificationMassesResult)
```

6.334 RangeManager

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/KERNEL/RangeManager.h>" namespace "OpenMS":

  cdef cppclass RangeManager1 "OpenMS::RangeManager<1>":
    # wrap-ignore
    RangeManager1()
    RangeManager1(RangeManager1)
    DPosition1 getMin()
    DPosition1 getMax()
    double getMinInt()
    double getMaxInt()
    void clearRanges()

  cdef cppclass RangeManager2 "OpenMS::RangeManager<2>":
    # wrap-ignore
    RangeManager2()
    RangeManager2(RangeManager2)
    DPosition2 getMin()
    DPosition2 getMax()
    double getMinInt()
    double getMaxInt()
    void clearRanges()
```

6.335 ReactionMonitoringTransition

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/MRM/ReactionMonitoringTransition.h>" namespace\
       "OpenMS":

  cdef cppclass ReactionMonitoringTransition:

    ReactionMonitoringTransition()
    ReactionMonitoringTransition(ReactionMonitoringTransition)    #wrap-ignore
    String getName()
    String getNativeID()
    String getPeptideRef()
    void setName(String name)
    void setNativeID(String name)
    void setPeptideRef(String peptide_ref)

    double getProductMZ()
    void setProductMZ(double)

    double getPrecursorMZ()
    void setPrecursorMZ(double)
```

```
    DecoyTransitionType getDecoyTransitionType()

    void setCompoundRef(String & compound_ref)
    String  getCompoundRef()
    void setPrecursorCVTermList(CVTermList & list_)
    void addPrecursorCVTerm(CVTerm & cv_term)
    CVTermList  getPrecursorCVTermList()
    void addProductCVTerm(CVTerm & cv_term)
    libcpp_vector[ TraMLProduct ]  getIntermediateProducts()
    void addIntermediateProduct(TraMLProduct product)
    void setIntermediateProducts(libcpp_vector[ TraMLProduct ] & products)
    void setProduct(TraMLProduct product)
    TraMLProduct  getProduct()
    void setRetentionTime(RetentionTime rt)
    RetentionTime  getRetentionTime()
    void setPrediction(Prediction & prediction)
    void addPredictionTerm(CVTerm & prediction)
    Prediction  getPrediction()
    void setDecoyTransitionType(DecoyTransitionType & d)
    double getLibraryIntensity()
    void setLibraryIntensity(double intensity)

    int getProductChargeState()
    bool isProductChargeStateSet()

    bool isDetectingTransition()
    void setDetectingTransition(bool val)

    bool isIdentifyingTransition()
    void setIdentifyingTransition(bool val)

    bool isQuantifyingTransition()
    void setQuantifyingTransition(bool val)

    void getKeys(libcpp_vector[String] & keys)
    void getKeys(libcpp_vector[unsigned int] & keys)
    DataValue getMetaValue(unsigned int)
    DataValue getMetaValue(String)
    void setMetaValue(unsigned int, DataValue)
    void setMetaValue(String, DataValue)
    bool metaValueExists(String)
    bool metaValueExists(unsigned int)
    void removeMetaValue(String)
    void removeMetaValue(unsigned int)

cdef extern from "<OpenMS/ANALYSIS/MRM/ReactionMonitoringTransition.h>" namespace\
       "OpenMS::ReactionMonitoringTransition":
```

```
  cdef enum DecoyTransitionType:

    UNKNOWN, TARGET, DECOY
```

6.336 RealMassDecomposer

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
ctypedef unsigned long long number_of_decompositions_type

cdef extern from "<OpenMS/CHEMISTRY/MASSDECOMPOSITION/IMS/RealMassDecomposer.h>"\
       namespace "OpenMS::ims":

  cdef cppclass RealMassDecomposer "OpenMS::ims::RealMassDecomposer":
    RealMassDecomposer()  #wrap-ignore
    RealMassDecomposer(RealMassDecomposer)  #wrap-ignore
    RealMassDecomposer(IMSWeights & weights)
    # libcpp_vector[ libcpp_vector[unsigned int] ] getDecompositions(double mass,\
       double error)
    # libcpp_vector[int] getDecompositions(double mass, double error,\
       constraints_type & constraints)
    unsigned long long getNumberOfDecompositions(double mass, double error)
```

6.337 Residue

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/Residue.h>" namespace "OpenMS":

  cdef cppclass Residue:
    # wrap-hash:
    #   getName().c_str()

    Residue()
    Residue(Residue)  # wrap-ignore

    Residue(String name,
        String three_letter_code,
        String one_letter_code,
        EmpiricalFormula formula)

    EmpiricalFormula getInternalToFull()
```

```
EmpiricalFormula getInternalToNTerm()
EmpiricalFormula getInternalToCTerm()
EmpiricalFormula getInternalToAIon()
EmpiricalFormula getInternalToBIon()
EmpiricalFormula getInternalToCIon()
EmpiricalFormula getInternalToXIon()
EmpiricalFormula getInternalToYIon()
EmpiricalFormula getInternalToZIon()

String getResidueTypeName(ResidueType res_type)

void setName(String name)

String getName()

void setShortName(String short_name)

String getShortName()

void setSynonyms(libcpp_set[String] synonyms)

void addSynonym(String synonym)

libcpp_set[String] getSynonyms()

void setThreeLetterCode(String three_letter_code)

String getThreeLetterCode()

void setOneLetterCode(String one_letter_code)

String getOneLetterCode()

void addLossFormula(EmpiricalFormula)

void setLossFormulas(libcpp_vector[EmpiricalFormula])

void addNTermLossFormula(EmpiricalFormula)

void setNTermLossFormulas(libcpp_vector[EmpiricalFormula])

libcpp_vector[EmpiricalFormula] getLossFormulas()

libcpp_vector[EmpiricalFormula] getNTermLossFormulas()

void setLossNames(libcpp_vector[String] name)

void setNTermLossNames(libcpp_vector[String] name)
```

```
void addLossName(String name)

void addNTermLossName(String name)

libcpp_vector[String] getLossNames()

libcpp_vector[String] getNTermLossNames()

void setFormula(EmpiricalFormula formula)

EmpiricalFormula getFormula(ResidueType res_type)

void setAverageWeight(double weight)

double getAverageWeight(ResidueType res_type)

void setMonoWeight(double weight)

double getMonoWeight(ResidueType res_type)

void setModification(String name)

String getModificationName()

void setLowMassIons(libcpp_vector[EmpiricalFormula] low_mass_ions)

libcpp_vector[EmpiricalFormula] getLowMassIons()

void setResidueSets(libcpp_set[String] residues_sets)

void addResidueSet(String residue_sets)

libcpp_set[String] getResidueSets()

bool hasNeutralLoss()

bool hasNTermNeutralLosses()

bool operator==(Residue & residue)

bool operator!=(Residue & residue)

bool operator==(char one_letter_code)

bool operator!=(char one_letter_code)

double getPka()
```

```
    double getPkb()

    double getPkc()

    double getPiValue()

    void setPka(double value)

    void setPkb(double value)

    void setPkc(double value)

    double getSideChainBasicity()

    void setSideChainBasicity(double gb_sc)

    double getBackboneBasicityLeft()

    void setBackboneBasicityLeft(double gb_bb_l)

    double getBackboneBasicityRight()

    void setBackboneBasicityRight(double gb_bb_r)

    bool isModified()

    bool isInResidueSet(String residue_set)

cdef extern from "<OpenMS/CHEMISTRY/Residue.h>" namespace "OpenMS::Residue":

  cdef enum ResidueType:
    # wrap-attach:
    #   Residue
    Full = 0,     # with N-terminus and C-terminus
    Internal,     # internal, without any termini
    NTerminal,    # only N-terminus
    CTerminal,    # only C-terminus
    AIon,       # N-terminus up to the C-alpha/carbonyl carbon bond
    BIon,       # N-terminus up to the peptide bond
    CIon,       # N-terminus up to the amide/C-alpha bond
    XIon,       # amide/C-alpha bond up to the C-terminus
    YIon,       # peptide bond up to the C-terminus
    ZIon,       # C-alpha/carbonyl carbon bond
    SizeOfResidueType
```

6.338 ResidueDB

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/ResidueDB.h>" namespace "OpenMS":

  cdef cppclass ResidueDB "OpenMS::ResidueDB":
    # wrap-manual-memory

    ResidueDB(ResidueDB)  #wrap-ignore

    Size getNumberOfResidues()
    Size getNumberOfModifiedResidues()
    const Residue * getResidue(String & name)
    const Residue * getModifiedResidue(String & name)
    const Residue * getModifiedResidue(Residue * residue, String & name)
    libcpp_set[ const Residue * ] getResidues(String & residue_set)
    libcpp_set[ String ] getResidueSets()
    void setResidues(String & filename)
    void addResidue(Residue & residue)
    bool hasResidue(String & name)
cdef extern from "<OpenMS/CHEMISTRY/ResidueDB.h>" namespace "OpenMS::ResidueDB":

  ResidueDB* getInstance()  # wrap-ignore
```

6.339 ResidueModification

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/ResidueModification.h>" namespace "OpenMS":

  cdef cppclass ResidueModification "OpenMS::ResidueModification":
    # wrap-hash:
    #   getFullId().c_str()

    ResidueModification()
    ResidueModification(ResidueModification)

    bool operator==(ResidueModification & modification)
    bool operator!=(ResidueModification & modification)

    void setId(String & id_)
    String  getId()
    void setFullId(String & full_id)
```

```
    String  getFullId()
    void setUniModAccession(String & id_)
    String  getUniModAccession()
    void setPSIMODAccession(String & id_)
    String  getPSIMODAccession()
    void setFullName(String & full_name)
    String  getFullName()
    void setName(String & name)
    String  getName()
    void setTermSpecificity(TermSpecificity term_spec)
    void setTermSpecificity(String & name)
    TermSpecificity getTermSpecificity()
    String getTermSpecificityName(TermSpecificity )
    void setOrigin(String & origin)
    String  getOrigin()
    void setSourceClassification(String & classification)
    void setSourceClassification(SourceClassification classification)
    SourceClassification getSourceClassification()
    String getSourceClassificationName(SourceClassification classification)
    void setAverageMass(double mass)
    double getAverageMass()
    void setMonoMass(double mass)
    double getMonoMass()
    void setDiffAverageMass(double mass)
    double getDiffAverageMass()
    void setDiffMonoMass(double mass)
    double getDiffMonoMass()
    void setFormula(String & composition)
    String  getFormula()
    void setDiffFormula(EmpiricalFormula & diff_formula)
    EmpiricalFormula  getDiffFormula()
    void setSynonyms(libcpp_set[ String ] & synonyms)
    void addSynonym(String & synonym)
    libcpp_set[ String ]  getSynonyms()
    void setNeutralLossDiffFormula(EmpiricalFormula & loss)
    EmpiricalFormula  getNeutralLossDiffFormula()
    void setNeutralLossMonoMass(double mono_mass)
    double getNeutralLossMonoMass()
    void setNeutralLossAverageMass(double average_mass)
    double getNeutralLossAverageMass()
    bool hasNeutralLoss()

cdef extern from "<OpenMS/CHEMISTRY/ResidueModification.h>" namespace\
       "OpenMS::ResidueModification":
  cdef enum TermSpecificity "OpenMS::ResidueModification::TermSpecificity":
    #wrap-attach:
    #  ResidueModification
    ANYWHERE
```

```
    C_TERM
    N_TERM
    PROTEIN_C_TERM
    PROTEIN_N_TERM
    NUMBER_OF_TERM_SPECIFICITY

cdef extern from "<OpenMS/CHEMISTRY/ResidueModification.h>" namespace\
       "OpenMS::ResidueModification":
  cdef enum SourceClassification\
       "OpenMS::ResidueModification::SourceClassification":
    #wrap-attach:
    #  ResidueModification
    ARTIFACT
    HYPOTHETICAL
    NATURAL
    POSTTRANSLATIONAL
    MULTIPLE
    CHEMICAL_DERIVATIVE
    ISOTOPIC_LABEL
    PRETRANSLATIONAL
    OTHER_GLYCOSYLATION
    NLINKED_GLYCOSYLATION
    AA_SUBSTITUTION
    OTHER
    NONSTANDARD_RESIDUE
    COTRANSLATIONAL
    OLINKED_GLYCOSYLATION
    UNKNOWN
    NUMBER_OF_SOURCE_CLASSIFICATIONS
```

6.340 RichPeak1D

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/KERNEL/RichPeak1D.h>" namespace "OpenMS":

  cdef cppclass RichPeak1D(Peak1D, MetaInfoInterface):
    # wrap-inherits:
    #  Peak1D
    #  MetaInfoInterface

    RichPeak1D()
    RichPeak1D(RichPeak1D &)
    RichPeak1D(float &, float &)
    bool operator==(RichPeak1D)
```

```
    bool operator!=(RichPeak1D)

    void getKeys(libcpp_vector[String] & keys)
    void getKeys(libcpp_vector[unsigned int] & keys)  # wrap-as:getKeysAsIntegers
    DataValue getMetaValue(unsigned int)
    DataValue getMetaValue(String)
    void setMetaValue(unsigned int, DataValue)
    void setMetaValue(String, DataValue)
    bool metaValueExists(String)
    bool metaValueExists(unsigned int)
    void removeMetaValue(String)
    void removeMetaValue(unsigned int)
```

6.341 RichPeak2D

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/KERNEL/RichPeak2D.h>" namespace "OpenMS":

  cdef cppclass RichPeak2D(Peak2D, UniqueIdInterface, MetaInfoInterface):
    # wrap-inherits:
    #  Peak2D
    #  UniqueIdInterface
    #  MetaInfoInterface

    RichPeak2D()
    RichPeak2D(RichPeak2D &)
    bool operator==(RichPeak2D)
    bool operator!=(RichPeak2D)

    void getKeys(libcpp_vector[String] & keys)
    void getKeys(libcpp_vector[unsigned int] & keys)  # wrap-as:getKeysAsIntegers
    DataValue getMetaValue(unsigned int)
    DataValue getMetaValue(String)
    void setMetaValue(unsigned int, DataValue)
    void setMetaValue(String, DataValue)
    bool metaValueExists(String)
    bool metaValueExists(unsigned int)
    void removeMetaValue(String)
    void removeMetaValue(unsigned int)
```

6.342 SILACLabeler

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/SIMULATION/LABELING/SILACLabeler.h>" namespace "OpenMS":

  cdef cppclass SILACLabeler :
    SILACLabeler()
    SILACLabeler(SILACLabeler)  #wrap-ignore
    void preCheck(Param & param)
    ## void setUpHook(SimTypes::FeatureMapSimVector & )
    ## void postDigestHook(SimTypes::FeatureMapSimVector & )
    ## void postRTHook(SimTypes::FeatureMapSimVector & )
    ## void postDetectabilityHook(SimTypes::FeatureMapSimVector & )
    ## void postIonizationHook(SimTypes::FeatureMapSimVector & )
    ## void postRawMSHook(SimTypes::FeatureMapSimVector & )
    ## void postRawTandemMSHook(SimTypes::FeatureMapSimVector & ,\
       SimTypes::MSSimExperiment & )
    ## # POINTER # BaseLabeler * create()
    String getProductName()
```

6.343 SVMWrapper

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/SVM/SVMWrapper.h>" namespace "OpenMS":

  cdef cppclass SVMWrapper "OpenMS::SVMWrapper":
    SVMWrapper()
    SVMWrapper(SVMWrapper)  #wrap-ignore
    void setParameter(SVM_parameter_type type_, Int value)
    void setParameter(SVM_parameter_type type_, double value)
    # Int train(struct svm_problem *problem)
    Int train(SVMData &problem)
    void saveModel(libcpp_string modelFilename)
    void loadModel(libcpp_string modelFilename)
    # void predict(struct svm_problem *problem, libcpp_vector[ double ]\
       &predicted_labels)
    void predict(SVMData &problem, libcpp_vector[ double ] &results)
    Int getIntParameter(SVM_parameter_type type_)
    double getDoubleParameter(SVM_parameter_type type_)
    # TODO STL map with wrapped key
    # void predict(libcpp_vector[ svm_node * ] &vectors, libcpp_vector[ double ]\
       &predicted_rts)
    # double performCrossValidation(svm_problem *problem_ul, SVMData &problem_l,\
       bool is_labeled, libcpp_map[ SVM_parameter_type, double ] &start_values_map,\
       libcpp_map[ SVM_parameter_type, double ] &step_sizes_map, libcpp_map[\
       SVM_parameter_type, double ] &end_values_map, Size number_of_partitions, Size\
       number_of_runs, libcpp_map[ SVM_parameter_type, double ] &best_parameters, bool\
```

```
        additive_step_sizesrue, bool outputalse, String, bool\
        mcc_as_performance_measurealse)
    double getSVRProbability()
    # void getSignificanceBorders(svm_problem *data, libcpp_pair[ double, double ]\
        &borders, double confidence.95, Size number_of_runs, Size number_of_partitions,\
        double step_size.01, Size max_iterations000000)
    void getSignificanceBorders(SVMData &data, libcpp_pair[ double, double ]\
        &sigmas, double confidence, Size number_of_runs, Size number_of_partitions, double\
        step_size, Size max_iterations)
    double getPValue(double sigma1, double sigma2, libcpp_pair[ double, double ]\
        point)
    # void getDecisionValues(svm_problem *data, libcpp_vector[ double ]\
        &decision_values)
    # void scaleData(svm_problem *data, Int max_scale_value1)
    # svm_problem * computeKernelMatrix(svm_problem *problem1, svm_problem\
        *problem2)
    # svm_problem * computeKernelMatrix(SVMData &problem1, SVMData &problem2)
    # void setTrainingSample(svm_problem *training_sample)
    void setTrainingSample(SVMData &training_sample)
    # void getSVCProbabilities(struct svm_problem *problem, libcpp_vector[ double ]\
        &probabilities, libcpp_vector[ double ] &prediction_labels)
    void setWeights(libcpp_vector[ int ] &weight_labels, libcpp_vector[ double ]\
        &weights)
    # void createRandomPartitions(svm_problem *problem, Size number, libcpp_vector[\
        svm_problem * ] &partitions)
    void createRandomPartitions(SVMData &problem, Size number, libcpp_vector[\
        SVMData ] &problems)
    # TODO: Mismatch between C++ return type ([u'svm_problem *']) and Python return\
        type (['void']) in function public mergePartitions:
    # svm_problem * mergePartitions(libcpp_vector[ svm_problem * ] &problems, Size\
        except)
    void mergePartitions(libcpp_vector[ SVMData ] &problems, Size except_, SVMData\
        &merged_problem)
    # void getLabels(svm_problem *problem, libcpp_vector[ double ] &labels)
    # double kernelOligo(libcpp_vector[ libcpp_pair[ int, double ] ] &x,\
        libcpp_vector[ libcpp_pair[ int, double ] ] &y, libcpp_vector[ double ]\
        &gauss_table, int max_distance1)
    # double kernelOligo(svm_node *x, svm_node *y, libcpp_vector[ double ]\
        &gauss_table, double sigma_square, Size max_distance0)
    void calculateGaussTable(Size border_length, double sigma, libcpp_vector[ double\
        ] &gauss_table)

cdef extern from "<OpenMS/ANALYSIS/SVM/SVMWrapper.h>" namespace "OpenMS":

  cdef cppclass SVMData "OpenMS::SVMData":
    SVMData()
    SVMData(SVMData)  #wrap-ignore
    # TODO nested STL
```

```
    # libcpp_vector[ libcpp_vector[ libcpp_pair[ Int, double ] ] ] sequences
    libcpp_vector[ double ] labels
    # TODO nested STL
    # SVMData(libcpp_vector[ libcpp_vector[ libcpp_pair[ Int, double ] ] ] &seqs,\
       libcpp_vector[ double ] &lbls)
    bool operator==(SVMData &rhs)
    bool store(String &filename)
    bool load(String &filename)

cdef extern from "<OpenMS/ANALYSIS/SVM/SVMWrapper.h>" namespace\
       "OpenMS::SVMWrapper":
  cdef enum SVM_parameter_type "OpenMS::SVMWrapper::SVM_parameter_type":
    #wrap-attach:
    #  SVMWrapper
    SVM_TYPE
    KERNEL_TYPE
    DEGREE
    C
    NU
    P
    GAMMA
    PROBABILITY
    SIGMA
    BORDER_LENGTH

cdef extern from "<OpenMS/ANALYSIS/SVM/SVMWrapper.h>" namespace\
       "OpenMS::SVMWrapper":
  cdef enum SVM_kernel_type "OpenMS::SVMWrapper::SVM_kernel_type":
    #wrap-attach:
    #  SVMWrapper
    OLIGO
    OLIGO_COMBINED
```

6.344 Sample

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/Sample.h>" namespace "OpenMS":

  cdef cppclass Sample(MetaInfoInterface):
    # wrap-inherits:
    #  MetaInfoInterface

    Sample()
    Sample(Sample)  # wrap-ignore
```

```
    String getName()
    void setName(String name)

    String getOrganism()
    void setOrganism(String organism)

    String getNumber()
    void setNumber(String number)

    String getComment()
    void setComment(String comment)

    SampleState getState()
    void setState(SampleState state)

    double getMass()
    void setMass(double mass)

    double getVolume()
    void setVolume(double volume)

    double getConcentration()
    void setConcentration(double concentration)

    libcpp_vector[Sample] getSubsamples()
    void setSubsamples(libcpp_vector[Sample] subsamples)

    void removeTreatment(UInt position)
    Int countTreatments()

cdef extern from "<OpenMS/METADATA/Sample.h>" namespace "OpenMS::Sample":

  cdef enum SampleState:
    # wrap-attach:
    #   Sample

    SAMPLENULL, SOLID, LIQUID, GAS, SOLUTION, EMULSION, SUSPENSION,\
       SIZE_OF_SAMPLESTATE
```

6.345 SampleTreatment

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/SampleTreatment.h>" namespace "OpenMS":
```

```
  cdef cppclass SampleTreatment(MetaInfoInterface) :
    # wrap-ignore
    # ABSTRACT class

    # wrap-inherits:
    #  MetaInfoInterface
    SampleTreatment(SampleTreatment)
    SampleTreatment(String & type_)

    bool operator==(SampleTreatment & rhs)
    String  getType()
    String  getComment()
    void setComment(String & comment)
```

6.346 SavitzkyGolayFilter

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/SMOOTHING/SavitzkyGolayFilter.h>" namespace\
       "OpenMS":

  cdef cppclass SavitzkyGolayFilter(DefaultParamHandler,ProgressLogger):
    # wrap-inherits:
    #  DefaultParamHandler
    #  ProgressLogger

    SavitzkyGolayFilter()
    SavitzkyGolayFilter(SavitzkyGolayFilter)

    void filter(MSSpectrum[Peak1D] & spectrum)
    void filterExperiment(MSExperiment[Peak1D,ChromatogramPeak] & exp)
```

6.347 Scaler

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/TRANSFORMERS/Scaler.h>" namespace "OpenMS":

  cdef cppclass Scaler(DefaultParamHandler):
    # wrap-inherits:
    #  DefaultParamHandler
```

```
    Scaler()
    Scaler(Scaler)  #wrap-ignore

    void filterSpectrum(MSSpectrum[Peak1D] & spec)
    void filterPeakSpectrum(MSSpectrum[Peak1D] & spec)
    void filterPeakMap(MSExperiment[Peak1D, ChromatogramPeak] & exp)
```

6.348 ScanWindow

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/ScanWindow.h>" namespace "OpenMS":

  cdef cppclass ScanWindow "OpenMS::ScanWindow":
    ScanWindow()
    ScanWindow(ScanWindow)
    double begin
    double end
```

6.349 SeedListGenerator

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
ctypedef libcpp_vector[DPosition2] SeedList

cdef extern from "<OpenMS/TRANSFORMATIONS/FEATUREFINDER/SeedListGenerator.h>"\
       namespace "OpenMS":

  cdef cppclass SeedListGenerator:

    SeedListGenerator()
    SeedListGenerator(SeedListGenerator &)

    void generateSeedList(MSExperiment[Peak1D, ChromatogramPeak] exp,\
       libcpp_vector[DPosition2] & seeds)
    void generateSeedList(libcpp_vector[PeptideIdentification] & peptides,\
       libcpp_vector[DPosition2] & seeds, bool use_peptide_mass)
    void generateSeedList(ConsensusMap & consensus, Map[unsigned long,\
       libcpp_vector[DPosition2] ] & seeds)   # wrap-ignore
    # TODO nested STL
    # void generateSeedLists(ConsensusMap & consensus, Map[ UInt64, libcpp_vector[\
       DPosition2] ] & seed_lists)
```

```
    void convertSeedList(libcpp_vector[ DPosition2] & seeds, FeatureMap & features)
    void convertSeedList(FeatureMap & features, libcpp_vector[ DPosition2] & seeds)
```

6.350 SemanticValidator

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/VALIDATORS/SemanticValidator.h>" namespace\
       "OpenMS::Internal":

  cdef cppclass SemanticValidator:

    SemanticValidator(CVMappings mapping, ControlledVocabulary cv)

    bool validate(String filename, StringList errors, StringList warnings)

    bool locateTerm(String path, SemanticValidator_CVTerm & parsed_term)

    void setTag(String tag)

    void setAccessionAttribute(String accession)

    void setNameAttribute(String name)

    void setValueAttribute(String value)

    void setCheckTermValueTypes(bool check)

    void setCheckUnits(bool check)

    void setUnitAccessionAttribute(String accession)

    void setUnitNameAttribute(String name)

cdef extern from "<OpenMS/FORMAT/VALIDATORS/SemanticValidator.h>" namespace\
       "OpenMS::Internal::SemanticValidator":

    cdef cppclass SemanticValidator_CVTerm\
       "OpenMS::Internal::SemanticValidator::CVTerm":
    SemanticValidator_CVTerm()
    SemanticValidator_CVTerm(SemanticValidator_CVTerm)
    String accession
    String name
    String value
    bool has_value
    String unit_accession
```

```
    bool has_unit_accession
    String unit_name
    bool has_unit_name
```

6.351 SequestInfile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/SequestInfile.h>" namespace "OpenMS":

  cdef cppclass SequestInfile "OpenMS::SequestInfile":
    SequestInfile()
    SequestInfile(SequestInfile)
    bool operator==(SequestInfile &sequest_infile)
    void store(String &filename)
    String getEnzymeInfoAsString()
    String  getDatabase()
    void setDatabase(String &database)
    String  getNeutralLossesForIons()
    void setNeutralLossesForIons(String &neutral_losses_for_ions)
    String  getIonSeriesWeights()
    void setIonSeriesWeights(String &ion_series_weights)
    String  getPartialSequence()
    void setPartialSequence(String &partial_sequence)
    String  getSequenceHeaderFilter()
    void setSequenceHeaderFilter(String &sequence_header_filter)
    String  getProteinMassFilter()
    void setProteinMassFilter(String &protein_mass_filter)
    float getPeakMassTolerance()
    void setPeakMassTolerance(float peak_mass_tolerance)
    float getPrecursorMassTolerance()
    void setPrecursorMassTolerance(float precursor_mass_tolerance)
    float getMatchPeakTolerance()
    void setMatchPeakTolerance(float match_peak_tolerance)
    float getIonCutoffPercentage()
    void setIonCutoffPercentage(float ion_cutoff_percentage)
    Size getPeptideMassUnit()
    void setPeptideMassUnit(Size peptide_mass_unit)
    Size getOutputLines()
    void setOutputLines(Size output_lines)
    Size getEnzymeNumber()
    String getEnzymeName()
    Size setEnzyme(String enzyme_name)
    Size getMaxAAPerModPerPeptide()
    void setMaxAAPerModPerPeptide(Size max_aa_per_mod_per_peptide)
    Size getMaxModsPerPeptide()
```

```
    void setMaxModsPerPeptide(Size max_mods_per_peptide)
    Size getNucleotideReadingFrame()
    void setNucleotideReadingFrame(Size nucleotide_reading_frame)
    Size getMaxInternalCleavageSites()
    void setMaxInternalCleavageSites(Size max_internal_cleavage_sites)
    Size getMatchPeakCount()
    void setMatchPeakCount(Size match_peak_count)
    Size getMatchPeakAllowedError()
    void setMatchPeakAllowedError(Size match_peak_allowed_error)
    bool getShowFragmentIons()
    void setShowFragmentIons(bool show_fragments)
    bool getPrintDuplicateReferences()
    void setPrintDuplicateReferences(bool print_duplicate_references)
    bool getRemovePrecursorNearPeaks()
    void setRemovePrecursorNearPeaks(bool remove_precursor_near_peaks)
    bool getMassTypeParent()
    void setMassTypeParent(bool mass_type_parent)
    bool getMassTypeFragment()
    void setMassTypeFragment(bool mass_type_fragment)
    bool getNormalizeXcorr()
    void setNormalizeXcorr(bool normalize_xcorr)
    bool getResiduesInUpperCase()
    void setResiduesInUpperCase(bool residues_in_upper_case)
    void addEnzymeInfo(libcpp_vector[ String ] &enzyme_info)
    libcpp_map[ String, libcpp_vector[ String ] ]  getModifications()  #\
       wrap-ignore
    void handlePTMs(String &modification_line, String &modifications_filename, bool\
       monoisotopic)
```

6.352 SequestOutfile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/SequestOutfile.h>" namespace "OpenMS":

  cdef cppclass SequestOutfile "OpenMS::SequestOutfile":
    SequestOutfile()
    SequestOutfile(SequestOutfile)
    bool operator==(SequestOutfile &sequest_outfile)
    void load(String &result_filename, libcpp_vector[ PeptideIdentification ]\
       &peptide_identifications, ProteinIdentification &protein_identification, double\
       p_value_threshold, libcpp_vector[ double ] &pvalues, String &database, bool\
       ignore_proteins_per_peptide)
    bool getColumns(String &line, libcpp_vector[ String ] &substrings, Size\
       number_of_columns, Size reference_column)
```

```
    void getSequences(String &database_filename,
              libcpp_map[ String, size_t ] &ac_position_map,
              libcpp_vector[ String ] &sequences,
              libcpp_vector[ libcpp_pair[ String, size_t ] ] &found,
              libcpp_map[ String, size_t ] &not_found)  # wrap-ignore
    void getACAndACType(String line, String &accession, String &accession_type)
    # TODO immutable types by reference
    # void readOutHeader(String &result_filename, DateTime &datetime, double\
       &precursor_mz_value, Int &charge, Size &precursor_mass_type, Size &ion_mass_type,\
       Size &displayed_peptides, String &sequest, String &sequest_version, String\
       &database_type, Int &number_column, Int &rank_sp_column, Int &id_column, Int\
       &mh_column, Int &delta_cn_column, Int &xcorr_column, Int &sp_column, Int\
       &sf_column, Int &ions_column, Int &reference_column, Int &peptide_column, Int\
       &score_column, Size &number_of_columns)
```

6.353 SignalToNoiseEstimator

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/NOISEESTIMATION/SignalToNoiseEstimator.h>"\
       namespace "OpenMS":

  cdef cppclass\
       SignalToNoiseEstimator[Container](DefaultParamHandler,ProgressLogger):
    # wrap-ignore
    # ABSTRACT class
    # wrap-inherits:
    #  DefaultParamHandler
    #  ProgressLogger
    SignalToNoiseEstimator()
    SignalToNoiseEstimator(SignalToNoiseEstimator)
    # void init(PeakIterator & it_begin, PeakIterator & it_end)
    # void init(Container & c)
    # double getSignalToNoise(PeakIterator & data_point)
    # double getSignalToNoise(Peak1D & data_point)
```

6.354 SignalToNoiseEstimatorMeanIterative

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from\
       "<OpenMS/FILTERING/NOISEESTIMATION/SignalToNoiseEstimatorMeanIterative.h>"\
```

```
      namespace "OpenMS":

  cdef cppclass SignalToNoiseEstimatorMeanIterative[Container](DefaultParamHandler,\
      ProgressLogger):
    # wrap-inherits:
    #  DefaultParamHandler
    #  ProgressLogger
    # wrap-instances:
    #   SignalToNoiseEstimator := SignalToNoiseEstimator[ MSSpectrum[Peak1D] ]

    SignalToNoiseEstimatorMeanIterative()
    SignalToNoiseEstimatorMeanIterative(SignalToNoiseEstimatorMeanIterative)
    void init(Container & c)
cdef extern from\
      "<OpenMS/FILTERING/NOISEESTIMATION/SignalToNoiseEstimatorMeanIterative.h>"\
      namespace "OpenMS::SignalToNoiseEstimatorMeanIterative<MSSpectrum<> >":

  cdef enum IntensityThresholdCalculation\
      "OpenMS::SignalToNoiseEstimatorMeanIterative<MSSpectrum<>\
      >::IntensityThresholdCalculation":
    MANUAL
    AUTOMAXBYSTDEV
    AUTOMAXBYPERCENT
```

6.355 SignalToNoiseEstimatorMedian

→ Link to OpenMS documentation

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/NOISEESTIMATION/SignalToNoiseEstimatorMedian.h>"\
      namespace "OpenMS":

  cdef cppclass SignalToNoiseEstimatorMedian[SpectrumT]:
    # wrap-instances:
    #   SignalToNoiseEstimatorMedian :=\
      SignalToNoiseEstimatorMedian[MSSpectrum[Peak1D]]

    SignalToNoiseEstimatorMedian()
    SignalToNoiseEstimatorMedian(SignalToNoiseEstimatorMedian)

    void init(MSSpectrum[Peak1D] & spectrum)
    void getSignalToNoise(Peak1D & data_point)

    double getSparseWindowPercent()
    double getHistogramRightmostPercent()

    # use wrap-ignore because autowrap cannot handle them at the moment
```

```
    # see addons/SignalToNoiseEstimatorMedianChrom.pyx for the implementation
    void init(MSChromatogram[ChromatogramPeak] & spectrum)  #wrap-ignore
    void getSignalToNoise(ChromatogramPeak & data_point)  #wrap-ignore

cdef extern from "<OpenMS/FILTERING/NOISEESTIMATION/SignalToNoiseEstimatorMedian.h>"\
       namespace "OpenMS::SignalToNoiseEstimatorMedian":

  cdef enum IntensityThresholdCalculation\
       "OpenMS::SignalToNoiseEstimatorMedianChrom::IntensityThresholdCalculation":
    MANUAL
    AUTOMAXBYSTDEV
    AUTOMAXBYPERCENT
```

6.356 SignalToNoiseEstimatorMedianRapid

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from\
       "<OpenMS/FILTERING/NOISEESTIMATION/SignalToNoiseEstimatorMedianRapid.h>" namespace\
       "OpenMS":

  cdef cppclass SignalToNoiseEstimatorMedianRapid\
       "OpenMS::SignalToNoiseEstimatorMedianRapid":

    SignalToNoiseEstimatorMedianRapid(SignalToNoiseEstimatorMedianRapid) \
       #wrap-ignore
    SignalToNoiseEstimatorMedianRapid(double window_length)
    NoiseEstimator estimateNoise(shared_ptr[Spectrum])
    NoiseEstimator estimateNoise(shared_ptr[Chromatogram])
    NoiseEstimator estimateNoise(libcpp_vector[ double ] mz_array, libcpp_vector[\
       double ] int_array)

cdef extern from\
       "<OpenMS/FILTERING/NOISEESTIMATION/SignalToNoiseEstimatorMedianRapid.h>" namespace\
       "OpenMS::SignalToNoiseEstimatorMedianRapid":

  cdef cppclass NoiseEstimator\
       "OpenMS::SignalToNoiseEstimatorMedianRapid::NoiseEstimator":

    NoiseEstimator()  #wrap-ignore
    NoiseEstimator(NoiseEstimator)  #wrap-ignore
    NoiseEstimator(double nr_windows_, double mz_start_, double win_len_)

    int nr_windows
    double mz_start
    double window_length
```

```
    libcpp_vector[ double ] result_windows_even
    libcpp_vector[ double ] result_windows_odd

    double get_noise_value(double mz)
    double get_noise_even(double mz)
    double get_noise_odd(double mz)
```

6.357 SimTypes

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/SIMULATION/SimTypes.h>" namespace "OpenMS::SimTypes":

  cdef cppclass SimRandomNumberGenerator:

    SimRandomNumberGenerator()
    SimRandomNumberGeneratorSimTypes(SimRandomNumberGeneratorSimTypes)  #\
       wrap-ignore
    void initialize(bool biological_random, bool technical_random)

  cdef cppclass SimProtein:
    SimProtein(FASTAEntry entry, MetaInfoInterface meta)
  ctypedef libcpp_vector[SimProtein] SampleProteins
  ctypedef libcpp_vector[libcpp_vector[SimProtein]] SampleChannels
```

6.358 SimpleOpenMSSpectraFactory

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from\
       "<OpenMS/ANALYSIS/OPENSWATH/DATAACCESS/SimpleOpenMSSpectraAccessFactory.h>"\
       namespace "OpenMS":

  cdef cppclass SimpleOpenMSSpectraFactory(ProgressLogger):
    # wrap-ignore

    SimpleOpenMSSpectraFactory()

    # shared_ptr[ISpectrumAccess]\
       getSpectrumAccessOpenMSPtr(MSExperiment[Peak1D,ChromatogramPeak] exp) #\
       wrap-ignore
    # OPENSWATHALGO_DLLAPI typedef boost::shared_ptr<ISpectrumAccess>\
       SpectrumAccessPtr;
```

6.359 SimplePairFinder

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/MAPMATCHING/SimplePairFinder.h>" namespace\
       "OpenMS":

  cdef cppclass SimplePairFinder(BaseGroupFinder) :
    # wrap-inherits:
    #  BaseGroupFinder

    SimplePairFinder()
    SimplePairFinder(SimplePairFinder)  #wrap-ignore

    void run(libcpp_vector[ ConsensusMap ] & input_maps, ConsensusMap & result_map)
    String getProductName()
```

6.360 Software

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/Software.h>" namespace "OpenMS":

  cdef cppclass Software:

    Software()
    Software(Software)  # wrap-ignore

    String getName()
    String getVersion()

    void setName(String)
    void setVersion(String)
```

6.361 SourceFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/SourceFile.h>" namespace "OpenMS":

  cdef cppclass SourceFile:
    SourceFile()
    SourceFile(SourceFile)
    libcpp_string getNameOfFile()
    void setNameOfFile(libcpp_string)

    libcpp_string getPathToFile()
    void setPathToFile(libcpp_string)

    float getFileSize()
    void setFileSize(float)

    libcpp_string getFileType()
    void setFileType(libcpp_string)

    libcpp_string getChecksum()
    void setChecksum(libcpp_string, ChecksumType)

    ChecksumType getChecksumType()

    libcpp_string getNativeIDType()
    void setNativeIDType(libcpp_string)

cdef extern from "<OpenMS/METADATA/SourceFile.h>" namespace "OpenMS::SourceFile":
  cdef enum ChecksumType:
       UNKNOWN_CHECKSUM, SHA1, MD5, SIZE_OF_CHECKSUMTYPE
```

6.362 SpectraMerger

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/TRANSFORMERS/SpectraMerger.h>" namespace\
       "OpenMS":

  cdef cppclass SpectraMerger(DefaultParamHandler):
    # wrap-inherits:
    #  DefaultParamHandler

    SpectraMerger()
    SpectraMerger(SpectraMerger)  #wrap-ignore

    void mergeSpectraBlockWise(MSExperiment[Peak1D, ChromatogramPeak] & exp)
    void mergeSpectraPrecursors(MSExperiment[Peak1D, ChromatogramPeak] & exp)
```

6.363 SpectraSTSimilarityScore

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/COMPARISON/SPECTRA/SpectraSTSimilarityScore.h>" namespace\
       "OpenMS":

  cdef cppclass SpectraSTSimilarityScore:
    SpectraSTSimilarityScore()
    SpectraSTSimilarityScore(SpectraSTSimilarityScore)
    bool preprocess(MSSpectrum[Peak1D] & spec, float\
       remove_peak_intensity_threshold, UInt cut_peaks_below, Size min_peak_number, Size\
       max_peak_number)
    BinnedSpectrum transform(MSSpectrum[Peak1D] & spec)
    double dot_bias(BinnedSpectrum & bin1, BinnedSpectrum & bin2, double\
       dot_product)
    double delta_D(double top_hit, double runner_up)
    double compute_F(double dot_product, double delta_D, double dot_bias)
    String getProductName()
```

6.364 SpectrumAccessOpenMS

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/OPENSWATH/DATAACCESS/SpectrumAccessOpenMS.h>"\
       namespace "OpenMS":

  cdef cppclass SpectrumAccessOpenMS(ISpectrumAccess):
    # wrap-inherits:
    #  ISpectrumAccess

    SpectrumAccessOpenMS() # wrap-pass-constructor

    SpectrumAccessOpenMS(SpectrumAccessOpenMS)
    SpectrumAccessOpenMS(shared_ptr[ MSExperiment[Peak1D, ChromatogramPeak] ] &\
       ms_experiment)
```

6.365 SpectrumAccessOpenMSCached

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from\
       "<OpenMS/ANALYSIS/OPENSWATH/DATAACCESS/SpectrumAccessOpenMSCached.h>" namespace\
       "OpenMS":

  cdef cppclass SpectrumAccessOpenMSCached(ISpectrumAccess):
    # wrap-inherits:
    #  ISpectrumAccess

    SpectrumAccessOpenMSCached() # wrap-pass-constructor

    SpectrumAccessOpenMSCached(String filename)
    SpectrumAccessOpenMSCached(SpectrumAccessOpenMSCached q)  # wrap-ignore
```

6.366 SpectrumAccessOpenMSInMemory

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from\
       "<OpenMS/ANALYSIS/OPENSWATH/DATAACCESS/SpectrumAccessOpenMSInMemory.h>" namespace\
       "OpenMS":

  cdef cppclass SpectrumAccessOpenMSInMemory(ISpectrumAccess) :
    # wrap-inherits:
    #  ISpectrumAccess

    SpectrumAccessOpenMSInMemory() # wrap-pass-constructor

    SpectrumAccessOpenMSInMemory(SpectrumAccessOpenMS)
    SpectrumAccessOpenMSInMemory(SpectrumAccessOpenMSCached)
    SpectrumAccessOpenMSInMemory(SpectrumAccessOpenMSInMemory)
    SpectrumAccessOpenMSInMemory(SpectrumAccessQuadMZTransforming)
```

6.367 SpectrumAccessQuadMZTransforming

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from\
       "<OpenMS/ANALYSIS/OPENSWATH/DATAACCESS/SpectrumAccessQuadMZTransforming.h>"\
       namespace "OpenMS":

  cdef cppclass SpectrumAccessQuadMZTransforming(SpectrumAccessTransforming) :
    # wrap-inherits:
```

```
    #  SpectrumAccessTransforming

    SpectrumAccessQuadMZTransforming()  # wrap-pass-constructor
    SpectrumAccessQuadMZTransforming(SpectrumAccessQuadMZTransforming) \
       #wrap-ignore

    SpectrumAccessQuadMZTransforming(shared_ptr[ SpectrumAccessOpenMS ], double a,\
       double b, double c, bool ppm)
    SpectrumAccessQuadMZTransforming(shared_ptr[ SpectrumAccessOpenMSCached ],\
       double a, double b, double c, bool ppm)
    SpectrumAccessQuadMZTransforming(shared_ptr[ SpectrumAccessOpenMSInMemory ],\
       double a, double b, double c, bool ppm)
```

6.368 SpectrumAccessTransforming

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from\
       "<OpenMS/ANALYSIS/OPENSWATH/DATAACCESS/SpectrumAccessTransforming.h>" namespace\
       "OpenMS":

  cdef cppclass SpectrumAccessTransforming(ISpectrumAccess) :
    # wrap-inherits:
    #  ISpectrumAccess
    # wrap-ignore
    # ABSTRACT class

    SpectrumAccessTransforming(SpectrumAccessTransforming)  # wrap-ignore
```

6.369 SpectrumAlignment

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/COMPARISON/SPECTRA/SpectrumAlignment.h>" namespace\
       "OpenMS":

  cdef cppclass SpectrumAlignment(DefaultParamHandler) :
    # wrap-inherits:
    #  DefaultParamHandler
    SpectrumAlignment()
    SpectrumAlignment(SpectrumAlignment)
```

```
    void getSpectrumAlignment(libcpp_vector[ libcpp_pair[ Size, Size ] ] &\
       alignment, MSSpectrum[Peak1D] & s1, MSSpectrum[Peak1D] & s2)   # wrap-ignore
    void getSpectrumAlignment(libcpp_vector[ libcpp_pair[ Size, Size ] ] &\
       alignment, MSSpectrum[RichPeak1D] & s1, MSSpectrum[RichPeak1D] & s2)   #\
       wrap-ignore
```

6.370 SpectrumAlignmentScore

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/COMPARISON/SPECTRA/SpectrumAlignmentScore.h>" namespace\
       "OpenMS":

  cdef cppclass SpectrumAlignmentScore(DefaultParamHandler):
    # wrap-inherits:
    #   DefaultParamHandler
    SpectrumAlignmentScore()
    SpectrumAlignmentScore(SpectrumAlignmentScore)

    double operator()(MSSpectrum[Peak1D] &, MSSpectrum[Peak1D] &)  #wrap-ignore
    double operator()(MSSpectrum[Peak1D] &)  #wrap-ignore
```

6.371 SpectrumIdentification

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/SpectrumIdentification.h>" namespace "OpenMS":

  cdef cppclass SpectrumIdentification(MetaInfoInterface):
    # wrap-inherits:
    #   MetaInfoInterface

    SpectrumIdentification()
    SpectrumIdentification(SpectrumIdentification)  # wrap-ignore

    void setHits(libcpp_vector[IdentificationHit] & hits)

    void addHit(IdentificationHit & hit)

    libcpp_vector[IdentificationHit] getHits()

    void getKeys(libcpp_vector[String] & keys)
    void getKeys(libcpp_vector[unsigned int] & keys)  # wrap-as:getKeysAsIntegers
```

```
    DataValue getMetaValue(unsigned int)
    DataValue getMetaValue(String)
    void setMetaValue(unsigned int, DataValue)
    void setMetaValue(String, DataValue)
    bool metaValueExists(String)
    bool metaValueExists(unsigned int)
    void removeMetaValue(String)
    void removeMetaValue(unsigned int)
```

6.372 SpectrumLookup

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/SpectrumLookup.h>" namespace "OpenMS":

  cdef cppclass SpectrumLookup:

    SpectrumLookup()
    double rt_tolerance

    bool empty()

    void readSpectra(MSExperiment[Peak1D, ChromatogramPeak] spectra, String\
       scan_regexp)

    Size findByRT(double rt)

    Size findByNativeID(String native_id)

    Size findByIndex(Size index, bool count_from_one)

    Size findByScanNumber(Size scan_number)

    Size findByReference(String spectrum_ref)

    void addReferenceFormat(String regexp)
```

6.373 SpectrumMetaDataLookup

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/SpectrumMetaDataLookup.h>" namespace "OpenMS":
```

```
  cdef cppclass SpectrumMetaDataLookup(SpectrumLookup):
    # wrap-inherits:
    #  SpectrumLookup

    SpectrumMetaDataLookup()
    void readSpectra(MSExperiment[Peak1D, ChromatogramPeak] spectra, String\
       scan_regexp, bool get_precursor_rt)

    void getSpectrumMetaData(Size index, SpectrumMetaData& meta)

    void getSpectrumMetaData(String spectrum_ref, SpectrumMetaData& meta)

    void getSpectrumMetaData(String spectrum_ref, SpectrumMetaData& meta, unsigned\
       char flags)

cdef extern from "<OpenMS/METADATA/SpectrumMetaDataLookup.h>" namespace\
       "OpenMS::SpectrumMetaDataLookup":

  void getSpectrumMetaData(MSSpectrum[Peak1D] spectrum,
               SpectrumMetaData& meta)  # wrap-attach:SpectrumMetaDataLookup

  bool addMissingRTsToPeptideIDs(libcpp_vector[PeptideIdentification],
                  String filename, bool stop_on_error)  #\
       wrap-attach:SpectrumMetaDataLookup

cdef extern from "<OpenMS/METADATA/SpectrumMetaDataLookup.h>" namespace\
       "OpenMS::SpectrumMetaDataLookup":

  cdef cppclass SpectrumMetaData\
       "OpenMS::SpectrumMetaDataLookup::SpectrumMetaData":

    SpectrumMetaData()
    SpectrumMetaData(SpectrumMetaData)  # wrap-ignore

    double rt
    double precursor_rt
    double precursor_mz
    Int precursor_charge
    Size ms_level
    Int scan_number
    String native_id
```

6.374 SpectrumSettings

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/SpectrumSettings.h>" namespace "OpenMS":

  cdef cppclass SpectrumSettings:

    SpectrumSettings()
    void unify(SpectrumSettings)
    int  getType()
    void setType(SpectrumType)
    String getNativeID()
    void setNativeID(String)
    String getComment()
    void setComment(String)

    InstrumentSettings getInstrumentSettings()
    void setInstrumentSettings(InstrumentSettings)

    AcquisitionInfo getAcquisitionInfo()
    void setAcquisitionInfo(AcquisitionInfo)

    SourceFile getSourceFile()
    void setSourceFile(SourceFile)

    libcpp_vector[Precursor] getPrecursors()
    void setPrecursors(libcpp_vector[Precursor])

    libcpp_vector[Product] getProducts()
    void setProducts(libcpp_vector[Product])

    libcpp_vector[PeptideIdentification] getPeptideIdentifications()
    void setPeptideIdentifications(libcpp_vector[PeptideIdentification])

    libcpp_vector[ shared_ptr[DataProcessing] ] getDataProcessing()
    void setDataProcessing(libcpp_vector[ shared_ptr[DataProcessing] ])

cdef extern from "<OpenMS/METADATA/SpectrumSettings.h>" namespace\
       "OpenMS::SpectrumSettings":

  cdef enum SpectrumType:
    # wrap-attach:
    #   SpectrumSettings
    UNKNOWN,  PEAKS, RAWDATA, SIZE_OF_SPECTRUMTYPE
```

6.375 Spline2d

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/MATH/MISC/Spline2d.h>" namespace "OpenMS":

  cdef cppclass Spline2d[ValType]:
    # wrap-instances:
    #   Spline2d := Spline2d[double]

    Spline2d()  # wrap-ignore
    Spline2d(Spline2d[ValType])  # wrap-ignore

    Spline2d(unsigned degree, libcpp_vector[ValType] x, libcpp_vector[double] y)

    ValType eval(ValType x)
    ValType derivatives(ValType x, unsigned order)
```

6.376 SplinePackage

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/DATAREDUCTION/SplinePackage.h>" namespace\
       "OpenMS":

  cdef cppclass SplinePackage "OpenMS::SplinePackage":

    SplinePackage(libcpp_vector[double] mz, libcpp_vector[double] intensity, double\
       scaling)
    SplinePackage(SplinePackage)  #wrap-ignore

    double getMzMin()
    double getMzMax()
    double getMzStepWidth()
    bool isInPackage(double mz)
    double eval(double mz)
```

6.377 SplineSpectrum

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/DATAREDUCTION/SplineSpectrum.h>" namespace\
       "OpenMS":

  cdef cppclass SplineSpectrum "OpenMS::SplineSpectrum":
```

```
    SplineSpectrum(libcpp_vector[double] mz, libcpp_vector[double] intensity)
    SplineSpectrum(libcpp_vector[double] mz, libcpp_vector[double] intensity, double\
       scaling)

    SplineSpectrum(MSSpectrum[Peak1D] raw_spectrum)
    SplineSpectrum(MSSpectrum[Peak1D] raw_spectrum, double scaling)

    double getMzMin()

    double getMzMax()

    int getSplineCount()
```

6.378 SqrtMower

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/TRANSFORMERS/SqrtMower.h>" namespace "OpenMS":

  cdef cppclass SqrtMower(DefaultParamHandler):
    # wrap-inherits:
    #  DefaultParamHandler

    SqrtMower()
    SqrtMower(SqrtMower)  #wrap-ignore

    void filterSpectrum(MSSpectrum[Peak1D] & spec)
    void filterPeakSpectrum(MSSpectrum[Peak1D] & spec)
    void filterPeakMap(MSExperiment[Peak1D, ChromatogramPeak] & exp)
```

6.379 StablePairFinder

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/MAPMATCHING/StablePairFinder.h>" namespace\
       "OpenMS":

  cdef cppclass StablePairFinder(BaseGroupFinder) :
    # wrap-inherits:
    #  BaseGroupFinder
    StablePairFinder()
    StablePairFinder(StablePairFinder)  #wrap-ignore
```

```
    void run(libcpp_vector[ ConsensusMap ] & input_maps, ConsensusMap & result_map)
    # POINTER # BaseGroupFinder * create()
    String getProductName()
```

6.380 String

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/DATASTRUCTURES/String.h>" namespace "OpenMS":

  cdef cppclass String:
    # wrap-hash:
    #   c_str()

    String()
    String(String)   # wrap-ignore
    String(char *)  # wrap-ignore
    String(str)  # wrap-ignore
    const_char * c_str()

    toString(self)  # wrap-ignore

    bool operator==(String)
    bool operator!=(String)

    size_t length()  # wrap-ignore
    # libcpp_string operator[](int)  # wrap-upper-limit:length()

cdef extern from "<OpenMS/DATASTRUCTURES/String.h>" namespace "OpenMS::String":

  cdef enum QuotingMethod "OpenMS::String::QuotingMethod":
    NONE
    ESCAPE
    DOUBLE
```

6.381 StringList

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
ctypedef libcpp_vector[ String ] StringList
```

6.382 SvmTheoreticalSpectrumGenerator

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/SvmTheoreticalSpectrumGenerator.h>" namespace\
       "OpenMS":

  cdef cppclass SvmTheoreticalSpectrumGenerator\
       "OpenMS::SvmTheoreticalSpectrumGenerator":
    SvmTheoreticalSpectrumGenerator()
    SvmTheoreticalSpectrumGenerator(SvmTheoreticalSpectrumGenerator)
    void load()
    libcpp_vector[ IonType ]  getIonTypes()

cdef extern from "<OpenMS/CHEMISTRY/SvmTheoreticalSpectrumGenerator.h>" namespace\
       "OpenMS::SvmTheoreticalSpectrumGenerator":

  cdef cppclass SvmModelParameterSet\
       "OpenMS::SvmTheoreticalSpectrumGenerator::SvmModelParameterSet":
    SvmModelParameterSet(SvmModelParameterSet)  #wrap-ignore
    # libcpp_vector[ shared_ptr[ SVMWrapper ] ] class_models
    # libcpp_vector[ shared_ptr[ SVMWrapper ] ] reg_models
    # TODO STL map with wrapped key
    # libcpp_map[ ResidueType, double ] _intensities
    # libcpp_vector[ IonType ] ion_types
    # TODO STL map with wrapped key
    # libcpp_map[ IonType, libcpp_vector[ IonType ] ] secondary_types
    Size number_intensity_levels
    Size number_regions
    libcpp_vector[ double ] feature_max
    libcpp_vector[ double ] feature_min
    double scaling_lower
    double scaling_upper
    libcpp_vector[ double ] intensity_bin_boarders
    libcpp_vector[ double ] intensity_bin_values
    # TODO nested STL
    # libcpp_map[ libcpp_pair[ IonType, Size ], libcpp_vector[ libcpp_vector[ double\
       ] ] ] conditional_prob

cdef extern from "<OpenMS/CHEMISTRY/SvmTheoreticalSpectrumGenerator.h>" namespace\
       "OpenMS::SvmTheoreticalSpectrumGenerator":

  cdef cppclass IonType "OpenMS::SvmTheoreticalSpectrumGenerator::IonType":
    IonType()
    IonType(IonType)
    IonType(ResidueType residue, EmpiricalFormula l, Int charge)
    bool operator<(IonType &rhs)
```

```
    ResidueType residue
    EmpiricalFormula loss
    Int charge

#     DescriptorSet(DescriptorSet)  #wrap-ignore
#     libcpp_vector[svm_node] descriptors # we would have to wrap libsvm for this
#
#
```

6.383 SvmTheoreticalSpectrumGeneratorSet

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/SvmTheoreticalSpectrumGeneratorSet.h>" namespace\
       "OpenMS":

  cdef cppclass SvmTheoreticalSpectrumGeneratorSet\
       "OpenMS::SvmTheoreticalSpectrumGeneratorSet":
    SvmTheoreticalSpectrumGeneratorSet()
    SvmTheoreticalSpectrumGeneratorSet(SvmTheoreticalSpectrumGeneratorSet)
    void load(String)
    void getSupportedCharges(libcpp_set[ size_t ] &charges)
    SvmTheoreticalSpectrumGenerator  getSvmModel(Size)
```

6.384 SvmTheoreticalSpectrumGeneratorTrainer

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/SvmTheoreticalSpectrumGeneratorTrainer.h>"\
       namespace "OpenMS":

  cdef cppclass SvmTheoreticalSpectrumGeneratorTrainer(DefaultParamHandler):
    # wrap-inherits:
    #  DefaultParamHandler

    SvmTheoreticalSpectrumGeneratorTrainer()
    SvmTheoreticalSpectrumGeneratorTrainer(SvmTheoreticalSpectrumGeneratorTrainer)

    void trainModel(MSExperiment[Peak1D,ChromatogramPeak] & spectra,\
       libcpp_vector[AASequence] & annotations, String filename, int precursor_charge)
    void normalizeIntensity(MSSpectrum[Peak1D] & S)
```

6.385 SwathFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/SwathFile.h>" namespace "OpenMS":

  cdef cppclass SwathFile(ProgressLogger) :
    # wrap-inherits:
    #  ProgressLogger
    SwathFile(SwathFile)  #wrap-ignore

    libcpp_vector[ SwathMap ] loadSplit(StringList file_list,
                      String tmp,
                      shared_ptr[ ExperimentalSettings ] exp_meta,
                      String readoptions)

    libcpp_vector[ SwathMap ] loadMzML(String file_,
                       String tmp,
                       shared_ptr[ ExperimentalSettings ] exp_meta,
                       String readoptions)

    libcpp_vector[ SwathMap ] loadMzXML(String file_,
                      String tmp,
                      shared_ptr[ ExperimentalSettings ] exp_meta,
                      String readoptions)
```

6.386 SwathFileConsumer

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/DATAACCESS/SwathFileConsumer.h>" namespace\
       "OpenMS":

  cdef cppclass FullSwathFileConsumer:
    #wrap-ignore

    FullSwathFileConsumer()  #wrap-ignore
    FullSwathFileConsumer(FullSwathFileConsumer)  #wrap-ignore
    FullSwathFileConsumer(libcpp_vector[SwathMap] swath_boundaries)

    void setExpectedSize(Size s, Size c)
    void setExperimentalSettings(ExperimentalSettings exp)

    void retrieveSwathMaps(libcpp_vector[SwathMap]& maps)
```

```
    void consumeSpectrum(MSSpectrum[Peak1D] & s)
    void consumeChromatogram(MSChromatogram[ChromatogramPeak] & c)

cdef extern from "<OpenMS/FORMAT/DATAACCESS/SwathFileConsumer.h>" namespace\
       "OpenMS":

  cdef cppclass RegularSwathFileConsumer(FullSwathFileConsumer):
    # wrap-inherits:
    #  FullSwathFileConsumer

    RegularSwathFileConsumer()
    RegularSwathFileConsumer(RegularSwathFileConsumer &)  # wrap-ignore

cdef extern from "<OpenMS/FORMAT/DATAACCESS/SwathFileConsumer.h>" namespace\
       "OpenMS":

  cdef cppclass CachedSwathFileConsumer(FullSwathFileConsumer):
    # wrap-inherits:
    #  FullSwathFileConsumer

    CachedSwathFileConsumer()  #wrap-ignore
    CachedSwathFileConsumer(CachedSwathFileConsumer &)  # wrap-ignore
    CachedSwathFileConsumer(String cachedir, String basename,
                Size nr_ms1_spectra, libcpp_vector[int] nr_ms2_spectra)
```

6.387 SwathMap

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/OPENSWATH/OPENSWATHALGO/DATAACCESS/SwathMap.h>"\
       namespace "OpenSwath":

  cdef cppclass SwathMap:

    SwathMap()
    SwathMap(SwathMap)

    double lower
    double upper
    double center
    bool ms1

    shared_ptr[ISpectrumAccess] sptr # wrap-ignore
```

6.388 SwathWindowLoader

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/OPENSWATH/SwathWindowLoader.h>" namespace\
       "OpenMS":

  cdef cppclass SwathWindowLoader:

    SwathWindowLoader()
    SwathWindowLoader(SwathWindowLoader)

    void annotateSwathMapsFromFile(libcpp_string filename,
                    libcpp_vector[ SwathMap ] & swath_maps, bool doSort)

    void readSwathWindows(libcpp_string filename,
                libcpp_vector[double] & swath_prec_lower_,
                libcpp_vector[double] & swath_prec_upper_ )
```

6.389 TICFilter

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/TRANSFORMERS/TICFilter.h>" namespace "OpenMS":

  cdef cppclass TICFilter(FilterFunctor) :
    # wrap-inherits:
    #  FilterFunctor
    TICFilter()
    TICFilter(TICFilter)
    double apply(MSSpectrum[Peak1D] & )
    # POINTER # FilterFunctor * create()
    String getProductName()
```

6.390 TMTSixPlexQuantitationMethod

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/QUANTITATION/TMTSixPlexQuantitationMethod.h>"\
       namespace "OpenMS":
```

```
  cdef cppclass TMTSixPlexQuantitationMethod(IsobaricQuantitationMethod) :
    # wrap-inherits:
    #  IsobaricQuantitationMethod
    TMTSixPlexQuantitationMethod()
    TMTSixPlexQuantitationMethod(TMTSixPlexQuantitationMethod)
```

6.391 TMTTenPlexQuantitationMethod

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/QUANTITATION/TMTTenPlexQuantitationMethod.h>"\
       namespace "OpenMS":

  cdef cppclass TMTTenPlexQuantitationMethod(IsobaricQuantitationMethod) :
    # wrap-inherits:
    #  IsobaricQuantitationMethod
    TMTTenPlexQuantitationMethod()
    TMTTenPlexQuantitationMethod(TMTTenPlexQuantitationMethod)
```

6.392 TOFCalibration

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/CALIBRATION/TOFCalibration.h>" namespace\
       "OpenMS":

  cdef cppclass TOFCalibration(DefaultParamHandler, ProgressLogger):
    # wrap-inherits:
    #  DefaultParamHandler
    #  ProgressLogger

    TOFCalibration()
    TOFCalibration(TOFCalibration)    #wrap-ignore

    void calibrate(MSExperiment[Peak1D, ChromatogramPeak] & input,\
       MSExperiment[Peak1D, ChromatogramPeak] & output, libcpp_vector[double] &\
       exp_masses)
    void pickAndCalibrate(MSExperiment[Peak1D, ChromatogramPeak] & input,\
       MSExperiment[Peak1D, ChromatogramPeak] & output, libcpp_vector[double] &\
       exp_masses)

    libcpp_vector[ double ]  getML1s()
```

```
    void setML1s(libcpp_vector[ double ] & ml1s)
    libcpp_vector[ double ]  getML2s()
    void setML2s(libcpp_vector[ double ] & ml2s)
    libcpp_vector[ double ]  getML3s()
    void setML3s(libcpp_vector[ double ] & ml3s)
```

6.393 Tagging

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/METADATA/Tagging.h>" namespace "OpenMS":

  cdef cppclass Tagging:

    Tagging()
    Tagging(Tagging)

    double getMassShift()
    void setMassShift(double mass_shift)

    IsotopeVariant getVariant()
    void setVariant(IsotopeVariant variant)

cdef extern from "<OpenMS/METADATA/Tagging.h>" namespace "OpenMS::Tagging":

  cdef enum IsotopeVariant "OpenMS::Tagging::IsotopeVariant":
    LIGHT
    HEAVY
    SIZE_OF_ISOTOPEVARIANT
```

6.394 TargetedExperiment

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/TARGETED/TargetedExperiment.h>" namespace\
       "OpenMS":

  cdef cppclass TargetedExperiment:

    TargetedExperiment()
    TargetedExperiment(TargetedExperiment &)
```

```
bool operator==(TargetedExperiment)
bool operator!=(TargetedExperiment)
TargetedExperiment operator+(TargetedExperiment)
TargetedExperiment iadd(TargetedExperiment)    # wrap-as:operator+=

void clear(bool clear_meta_data)
void sortTransitionsByProductMZ()

void setCVs(libcpp_vector[CV] cvs)
libcpp_vector[CV] getCVs()
void addCV(CV cv)

void setContacts(libcpp_vector[Contact] contacts)
libcpp_vector[Contact] getContacts()
void addContact(Contact contact)

void setPublications(libcpp_vector[Publication] publications)
libcpp_vector[Publication] getPublications()
void addPublication(Publication publication)

void setTargetCVTerms(CVTermList cv_terms)
CVTermList getTargetCVTerms()
void addTargetCVTerm(CVTerm cv_term)
void setTargetMetaValue(String name, DataValue value)

void setInstruments(libcpp_vector[TargetedExperiment_Instrument] instruments)
libcpp_vector[TargetedExperiment_Instrument] getInstruments()
void addInstrument(TargetedExperiment_Instrument instrument)

void setSoftware(libcpp_vector[Software] software)
libcpp_vector[Software] getSoftware()
void addSoftware(Software software)

void setProteins(libcpp_vector[Protein] proteins)
libcpp_vector[Protein] getProteins()
Protein getProteinByRef(String ref)
void addProtein(Protein protein)

void setCompounds(libcpp_vector[Compound] rhs)
libcpp_vector[Compound] getCompounds()
void addCompound(Compound rhs)

void setPeptides(libcpp_vector[Peptide] rhs)
libcpp_vector[Peptide] getPeptides()
Peptide getPeptideByRef(String ref)
Compound getCompoundByRef(String ref)
void addPeptide(Peptide rhs)
```

```
    void setTransitions(libcpp_vector[ReactionMonitoringTransition] transitions)
    libcpp_vector[ReactionMonitoringTransition] getTransitions()
    void addTransition(ReactionMonitoringTransition transition)

    void setIncludeTargets(libcpp_vector[IncludeExcludeTarget] targets)
    libcpp_vector[IncludeExcludeTarget] getIncludeTargets()
    void addIncludeTarget(IncludeExcludeTarget target)
    void setExcludeTargets(libcpp_vector[IncludeExcludeTarget] targets)
    libcpp_vector[IncludeExcludeTarget] getExcludeTargets()
    void addExcludeTarget(IncludeExcludeTarget target)

    void setSourceFiles(libcpp_vector[SourceFile] source_files)
    libcpp_vector[SourceFile] getSourceFiles()
    void addSourceFile(SourceFile source_file)

    bool containsInvalidReferences()
```

6.395 TargetedExperimentHelper

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/TARGETED/TargetedExperimentHelper.h>" namespace\
       "OpenMS::TargetedExperimentHelper":

  cdef cppclass Configuration :
    Configuration(Configuration)  #wrap-ignore
    String contact_ref
    String instrument_ref
    libcpp_vector[ CVTermList ] validations

    void setCVTerms(libcpp_vector[CVTerm] & terms)
    void replaceCVTerm(CVTerm & term)

    void replaceCVTerms(libcpp_vector[CVTerm] cv_terms,
              String accession
               )

    void replaceCVTerms(Map[String, libcpp_vector[CVTerm] ] cv_term_map
               )

    Map[String, libcpp_vector[CVTerm] ] getCVTerms()
    void addCVTerm(CVTerm & term)

    bool hasCVTerm(String accession)
    bool empty()
```

```
  void getKeys(libcpp_vector[String] & keys)
  void getKeys(libcpp_vector[unsigned int] & keys)  # wrap-as:getKeysAsIntegers
  DataValue getMetaValue(unsigned int)
  DataValue getMetaValue(String)
  void setMetaValue(unsigned int, DataValue)
  void setMetaValue(String, DataValue)
  bool metaValueExists(String)
  bool metaValueExists(unsigned int)
  void removeMetaValue(String)
  void removeMetaValue(unsigned int)

cdef cppclass CV:
  CV(CV)
  CV(String new_id, String new_fullname, String new_version, String new_URI)

  String id
  String fullname
  String version
  String URI

cdef cppclass Protein:
  Protein()
  Protein(Protein)

  String id
  String sequence

  void setCVTerms(libcpp_vector[CVTerm] & terms)
  void replaceCVTerm(CVTerm & term)

  void replaceCVTerms(libcpp_vector[CVTerm] cv_terms,
            String accession
             )

  void replaceCVTerms(Map[String, libcpp_vector[CVTerm] ] cv_term_map
             )

  Map[String, libcpp_vector[CVTerm] ] getCVTerms()
  void addCVTerm(CVTerm & term)

  bool hasCVTerm(String accession)
  bool empty()

  void getKeys(libcpp_vector[String] & keys)
  void getKeys(libcpp_vector[unsigned int] & keys)  # wrap-as:getKeysAsIntegers
  DataValue getMetaValue(unsigned int)
  DataValue getMetaValue(String)
```

```
  void setMetaValue(unsigned int, DataValue)
  void setMetaValue(String, DataValue)
  bool metaValueExists(String)
  bool metaValueExists(unsigned int)
  void removeMetaValue(String)
  void removeMetaValue(unsigned int)

cdef cppclass RetentionTime:
  RetentionTime()
  RetentionTime(RetentionTime)
  String software_ref

  void setCVTerms(libcpp_vector[CVTerm] & terms)
  void replaceCVTerm(CVTerm & term)

  void replaceCVTerms(libcpp_vector[CVTerm] cv_terms,
            String accession
             )

  void replaceCVTerms(Map[String, libcpp_vector[CVTerm] ] cv_term_map
             )

  Map[String, libcpp_vector[CVTerm] ] getCVTerms()
  void addCVTerm(CVTerm & term)

  bool hasCVTerm(String accession)
  bool empty()

  void getKeys(libcpp_vector[String] & keys)
  void getKeys(libcpp_vector[unsigned int] & keys)  # wrap-as:getKeysAsIntegers
  DataValue getMetaValue(unsigned int)
  DataValue getMetaValue(String)
  void setMetaValue(unsigned int, DataValue)
  void setMetaValue(String, DataValue)
  bool metaValueExists(String)
  bool metaValueExists(unsigned int)
  void removeMetaValue(String)
  void removeMetaValue(unsigned int)

cdef cppclass Compound :
  Compound()
  Compound(Compound)
  bool operator==(Compound & rhs)

  String id
  String molecular_formula
  String smiles_string
  double theoretical_mass
```

```
    libcpp_vector[ RetentionTime ] rts

    void setChargeState(int charge)
    bool hasCharge()
    int getChargeState()

    void setCVTerms(libcpp_vector[CVTerm] & terms)
    void replaceCVTerm(CVTerm & term)

    void replaceCVTerms(libcpp_vector[CVTerm] cv_terms,
              String accession
               )

    void replaceCVTerms(Map[String, libcpp_vector[CVTerm] ] cv_term_map
               )

    Map[String, libcpp_vector[CVTerm] ] getCVTerms()
    void addCVTerm(CVTerm & term)

    bool hasCVTerm(String accession)
    bool empty()

    void getKeys(libcpp_vector[String] & keys)
    void getKeys(libcpp_vector[unsigned int] & keys)  # wrap-as:getKeysAsIntegers
    DataValue getMetaValue(unsigned int)
    DataValue getMetaValue(String)
    void setMetaValue(unsigned int, DataValue)
    void setMetaValue(String, DataValue)
    bool metaValueExists(String)
    bool metaValueExists(unsigned int)
    void removeMetaValue(String)
    void removeMetaValue(unsigned int)

  cdef cppclass Peptide:
    Peptide()
    Peptide(Peptide)

    libcpp_vector[RetentionTime] rts
    String id
    libcpp_vector[String] protein_refs
    CVTermList evidence
    String sequence
    libcpp_vector[TargetedExperiment_Modification] mods

    void setChargeState(int charge)
    int getChargeState()
    bool hasCharge()
    void setPeptideGroupLabel(String label)
```

```
  String getPeptideGroupLabel()
  double getRetentionTime()

  void setCVTerms(libcpp_vector[CVTerm] & terms)
  void replaceCVTerm(CVTerm & term)

  void replaceCVTerms(libcpp_vector[CVTerm] cv_terms,
            String accession
             )

  void replaceCVTerms(Map[String, libcpp_vector[CVTerm] ] cv_term_map
             )

  Map[String, libcpp_vector[CVTerm] ] getCVTerms()
  void addCVTerm(CVTerm & term)

  bool hasCVTerm(String accession)
  bool empty()

  void getKeys(libcpp_vector[String] & keys)
  void getKeys(libcpp_vector[unsigned int] & keys)  # wrap-as:getKeysAsIntegers
  DataValue getMetaValue(unsigned int)
  DataValue getMetaValue(String)
  void setMetaValue(unsigned int, DataValue)
  void setMetaValue(String, DataValue)
  bool metaValueExists(String)
  bool metaValueExists(unsigned int)
  void removeMetaValue(String)
  void removeMetaValue(unsigned int)

cdef cppclass Contact :
  Contact()
  Contact(Contact)  #wrap-ignore
  String id
  bool operator==(Contact & rhs)

  void setCVTerms(libcpp_vector[CVTerm] & terms)
  void replaceCVTerm(CVTerm & term)

  void replaceCVTerms(libcpp_vector[CVTerm] cv_terms,
            String accession
             )

  void replaceCVTerms(Map[String, libcpp_vector[CVTerm] ] cv_term_map
             )

  Map[String, libcpp_vector[CVTerm] ] getCVTerms()
  void addCVTerm(CVTerm & term)
```

```
  bool hasCVTerm(String accession)
  bool empty()

  void getKeys(libcpp_vector[String] & keys)
  void getKeys(libcpp_vector[unsigned int] & keys)  # wrap-as:getKeysAsIntegers
  DataValue getMetaValue(unsigned int)
  DataValue getMetaValue(String)
  void setMetaValue(unsigned int, DataValue)
  void setMetaValue(String, DataValue)
  bool metaValueExists(String)
  bool metaValueExists(unsigned int)
  void removeMetaValue(String)
  void removeMetaValue(unsigned int)

cdef cppclass Publication :
  Publication()
  Publication(Publication)  #wrap-ignore
  String id
  bool operator==(Publication & rhs)

  void setCVTerms(libcpp_vector[CVTerm] & terms)
  void replaceCVTerm(CVTerm & term)

  void replaceCVTerms(libcpp_vector[CVTerm] cv_terms,
            String accession
             )

  void replaceCVTerms(Map[String, libcpp_vector[CVTerm] ] cv_term_map
             )

  Map[String, libcpp_vector[CVTerm] ] getCVTerms()
  void addCVTerm(CVTerm & term)

  bool hasCVTerm(String accession)
  bool empty()

  void getKeys(libcpp_vector[String] & keys)
  void getKeys(libcpp_vector[unsigned int] & keys)  # wrap-as:getKeysAsIntegers
  DataValue getMetaValue(unsigned int)
  DataValue getMetaValue(String)
  void setMetaValue(unsigned int, DataValue)
  void setMetaValue(String, DataValue)
  bool metaValueExists(String)
  bool metaValueExists(unsigned int)
  void removeMetaValue(String)
  void removeMetaValue(unsigned int)
```

```
cdef cppclass TargetedExperiment_Instrument\
     "OpenMS::TargetedExperimentHelper::Instrument":
  TargetedExperiment_Instrument()
  TargetedExperiment_Instrument(TargetedExperiment_Instrument)  #wrap-ignore
  String id
  bool operator==(TargetedExperiment_Instrument & rhs)

  void setCVTerms(libcpp_vector[CVTerm] & terms)
  void replaceCVTerm(CVTerm & term)

  void replaceCVTerms(libcpp_vector[CVTerm] cv_terms,
            String accession
             )

  void replaceCVTerms(Map[String, libcpp_vector[CVTerm] ] cv_term_map
             )

  Map[String, libcpp_vector[CVTerm] ] getCVTerms()
  void addCVTerm(CVTerm & term)

  bool hasCVTerm(String accession)
  bool empty()

  void getKeys(libcpp_vector[String] & keys)
  void getKeys(libcpp_vector[unsigned int] & keys)  # wrap-as:getKeysAsIntegers
  DataValue getMetaValue(unsigned int)
  DataValue getMetaValue(String)
  void setMetaValue(unsigned int, DataValue)
  void setMetaValue(String, DataValue)
  bool metaValueExists(String)
  bool metaValueExists(unsigned int)
  void removeMetaValue(String)
  void removeMetaValue(unsigned int)

cdef cppclass Prediction :
  Prediction()
  Prediction(Prediction)  #wrap-ignore
  bool operator==(Prediction & rhs)

  String software_ref
  String contact_ref

  void setCVTerms(libcpp_vector[CVTerm] & terms)
  void replaceCVTerm(CVTerm & term)

  void replaceCVTerms(libcpp_vector[CVTerm] cv_terms,
            String accession
             )
```

```
    void replaceCVTerms(Map[String, libcpp_vector[CVTerm] ] cv_term_map
                )

    Map[String, libcpp_vector[CVTerm] ] getCVTerms()
    void addCVTerm(CVTerm & term)

    bool hasCVTerm(String accession)
    bool empty()

    void getKeys(libcpp_vector[String] & keys)
    void getKeys(libcpp_vector[unsigned int] & keys)  # wrap-as:getKeysAsIntegers
    DataValue getMetaValue(unsigned int)
    DataValue getMetaValue(String)
    void setMetaValue(unsigned int, DataValue)
    void setMetaValue(String, DataValue)
    bool metaValueExists(String)
    bool metaValueExists(unsigned int)
    void removeMetaValue(String)
    void removeMetaValue(unsigned int)

  cdef cppclass TargetedExperiment_Interpretation\
      "OpenMS::TargetedExperimentHelper::Interpretation":
    TargetedExperiment_Interpretation()
    TargetedExperiment_Interpretation(TargetedExperiment_Interpretation) \
      #wrap-ignore

    unsigned char ordinal
    unsigned char rank
    ResidueType iontype

    void setCVTerms(libcpp_vector[CVTerm] & terms)
    void replaceCVTerm(CVTerm & term)

    void replaceCVTerms(Map[String, libcpp_vector[CVTerm] ] cv_term_map
                )

    Map[String, libcpp_vector[CVTerm] ] getCVTerms()
    void addCVTerm(CVTerm & term)

    bool hasCVTerm(String accession)
    bool empty()

    void getKeys(libcpp_vector[String] & keys)
    void getKeys(libcpp_vector[unsigned int] & keys)  # wrap-as:getKeysAsIntegers
    DataValue getMetaValue(unsigned int)
    DataValue getMetaValue(String)
    void setMetaValue(unsigned int, DataValue)
```

```
    void setMetaValue(String, DataValue)
    bool metaValueExists(String)
    bool metaValueExists(unsigned int)
    void removeMetaValue(String)
    void removeMetaValue(unsigned int)

  cdef cppclass TraMLProduct :
    TraMLProduct()
    TraMLProduct(TraMLProduct)  #wrap-ignore
    bool operator==(TraMLProduct & rhs)

    void setChargeState(int charge)
    int getChargeState()
    bool hasCharge()
    libcpp_vector[ Configuration ]  getConfigurationList()
    void addConfiguration(Configuration configuration)
    libcpp_vector[ TargetedExperiment_Interpretation ] getInterpretationList()
    void addInterpretation(TargetedExperiment_Interpretation interpretation)
    void resetInterpretations()

    void setCVTerms(libcpp_vector[CVTerm] & terms)
    void replaceCVTerm(CVTerm & term)

    void replaceCVTerms(Map[String, libcpp_vector[CVTerm] ] cv_term_map
                )

    Map[String, libcpp_vector[CVTerm] ] getCVTerms()
    void addCVTerm(CVTerm & term)

    bool hasCVTerm(String accession)
    bool empty()

    void getKeys(libcpp_vector[String] & keys)
    void getKeys(libcpp_vector[unsigned int] & keys)  # wrap-as:getKeysAsIntegers
    DataValue getMetaValue(unsigned int)
    DataValue getMetaValue(String)
    void setMetaValue(unsigned int, DataValue)
    void setMetaValue(String, DataValue)
    bool metaValueExists(String)
    bool metaValueExists(unsigned int)
    void removeMetaValue(String)
    void removeMetaValue(unsigned int)

cdef extern from "<OpenMS/ANALYSIS/TARGETED/TargetedExperimentHelper.h>" namespace\
       "OpenMS::TargetedExperimentHelper::Peptide":

  cdef cppclass TargetedExperiment_Modification\
       "OpenMS::TargetedExperimentHelper::Peptide::Modification":
```

```
    TargetedExperiment_Modification()
    TargetedExperiment_Modification(TargetedExperiment_Modification)

    double avg_mass_delta
    int location
    double mono_mass_delta
```

6.396 TextFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/TextFile.h>" namespace "OpenMS":

  cdef cppclass TextFile "OpenMS::TextFile":
    TextFile()
    TextFile(TextFile)  #wrap-ignore
    TextFile(String &filename, bool trim_linesalse, Int first_n1)
    void load(String &filename, bool trim_linesalse, Int first_n1)
    void store(String &filename)
    void addLine(String line)
```

6.397 TheoreticalSpectrumGenerator

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/TheoreticalSpectrumGenerator.h>" namespace\
       "OpenMS":

  cdef cppclass TheoreticalSpectrumGenerator(DefaultParamHandler):
    # wrap-inherits:
    #  DefaultParamHandler

    TheoreticalSpectrumGenerator()
    TheoreticalSpectrumGenerator(TheoreticalSpectrumGenerator)
    void getSpectrum(MSSpectrum[RichPeak1D] &spec, AASequence &peptide, Int charge)
    void addPeaks(MSSpectrum[RichPeak1D] &spectrum, AASequence &peptide, ResidueType\
       res_type, Int charge)
    void addPrecursorPeaks(MSSpectrum[RichPeak1D] &spec, AASequence &peptide, Int\
       charge)
    void addAbundantImmoniumIons(MSSpectrum[RichPeak1D] &spec, AASequence &peptide)
```

6.398 ThresholdMower

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/TRANSFORMERS/ThresholdMower.h>" namespace\
       "OpenMS":

  cdef cppclass ThresholdMower(DefaultParamHandler):
    # wrap-inherits:
    #  DefaultParamHandler

    ThresholdMower()
    ThresholdMower(ThresholdMower)  #wrap-ignore

    void filterSpectrum(MSSpectrum[Peak1D] & spec)
    void filterPeakSpectrum(MSSpectrum[Peak1D] & spec)
    void filterPeakMap(MSExperiment[Peak1D, ChromatogramPeak] & exp)
```

6.399 TraMLFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/TraMLFile.h>" namespace "OpenMS":

  cdef cppclass TraMLFile:

    TraMLFile()

    void load(String filename,
          TargetedExperiment & id)

    void store(String filename,
          TargetedExperiment & id)

    bool isSemanticallyValid(String filename, StringList & errors,
                 StringList & warnings)
```

6.400 TransformationDescription

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/MAPMATCHING/TransformationDescription.h>"\
       namespace "OpenMS":

  cdef cppclass TransformationDescription:
    TransformationDescription()
    TransformationDescription(TransformationDescription)  # wrap-ignore
    libcpp_vector[libcpp_pair[double,double]] getDataPoints()
    void setDataPoints(libcpp_vector[libcpp_pair[double,double]] & data)
    double apply(double)

    void fitModel(String model_type, Param params)
    void fitModel(String model_type)

    String getModelType()
    Param getModelParameters()

    void invert()

cdef extern from "<OpenMS/ANALYSIS/MAPMATCHING/TransformationDescription.h>"\
       namespace "OpenMS::TransformationDescription":

  void getModelTypes(StringList result)  # wrap-attach:TransformationDescription
```

6.401 TransformationModel

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/MAPMATCHING/TransformationModel.h>" namespace\
       "OpenMS":

  cdef cppclass TransformationModel:
    # wrap-ignore

    TransformationModel()
    TransformationModel(TransformationModel, Param)  # wrap-ignore

    Param getParameters()
```

6.402 TransformationModelBSpline

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/MAPMATCHING/TransformationModelBSpline.h>"\
       namespace "OpenMS":

  cdef cppclass TransformationModelBSpline(TransformationModel) :
    # wrap-inherits:
    #  TransformationModel

    TransformationModelBSpline(TransformationModelBSpline)  #wrap-ignore

    TransformationModelBSpline(libcpp_vector[ libcpp_pair[double, double ] ] data,\
       Param params)

    double evaluate(double value)

cdef extern from "<OpenMS/ANALYSIS/MAPMATCHING/TransformationModelBSpline.h>"\
       namespace "OpenMS::TransformationModelBSpline":

  void getDefaultParameters(Param & params)   #\
       wrap-attach:TransformationModelBSpline
```

6.403 TransformationModelInterpolated

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/MAPMATCHING/TransformationModelInterpolated.h>"\
       namespace "OpenMS":

  cdef cppclass TransformationModelInterpolated(TransformationModel):
    # wrap-inherits:
    #   TransformationModel

    TransformationModelInterpolated(TransformationModelInterpolated)  #wrap-ignore
    TransformationModelInterpolated(libcpp_vector[ libcpp_pair[double, double] ]&\
       data, Param & params)

    void getDefaultParameters(Param &)
    double evaluate(double value)

cdef extern from "<OpenMS/ANALYSIS/MAPMATCHING/TransformationModelInterpolated.h>"\
       namespace "OpenMS:TransformationModelInterpolated":

  cdef cppclass Interpolator:
    # wrap-ignore
    # ABSTRACT

    void init(libcpp_vector[double] x, libcpp_vector[double] y)
```

```
    double eval(double x)
```

6.404 TransformationModelLinear

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/MAPMATCHING/TransformationModelLinear.h>"\
       namespace "OpenMS":

  cdef cppclass TransformationModelLinear(TransformationModel):
    # wrap-inherits:
    #   TransformationModel

    TransformationModelLinear(TransformationModelLinear)  #wrap-ignore
    TransformationModelLinear(libcpp_vector[ libcpp_pair[double, double] ]& data,\
       Param & params)

    void getDefaultParameters(Param &)

    double evaluate(double value)
    void getParameters(double & slope, double & intercept)
    void invert()
```

6.405 TransformationModelLowess

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/MAPMATCHING/TransformationModelLowess.h>"\
       namespace "OpenMS":

  cdef cppclass TransformationModelLowess(TransformationModel) :
    # wrap-inherits:
    #  TransformationModel

    TransformationModelLowess()  #wrap-ignore
    TransformationModelLowess(TransformationModelLowess)  #wrap-ignore
    TransformationModelLowess(libcpp_vector[ libcpp_pair[double, double] ]& data,\
       Param & params)

    double evaluate(double value)
    void getDefaultParameters(Param & params)
```

6.406 TransformationXMLFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/TransformationXMLFile.h>" namespace "OpenMS":

  cdef cppclass TransformationXMLFile:
    TransformationXMLFile()

    void load(String, TransformationDescription &, bool fix_model) nogil except+
    void store(String, TransformationDescription) nogil except+
```

6.407 TransitionTSVReader

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/ANALYSIS/OPENSWATH/TransitionTSVReader.h>" namespace\
       "OpenMS":

  cdef cppclass TransitionTSVReader(ProgressLogger):
    # wrap-inherits:
    #  ProgressLogger

    TransitionTSVReader()
    TransitionTSVReader(TransitionTSVReader)   # wrap-ignore

    void convertTargetedExperimentToTSV(char * filename, TargetedExperiment&\
       targeted_exp)

    void convertTSVToTargetedExperiment(char * filename, Type filetype,\
       TargetedExperiment& targeted_exp)

    void convertTSVToTargetedExperiment(char * filename, Type filetype,\
       LightTargetedExperiment& targeted_exp)

    void validateTargetedExperiment(TargetedExperiment targeted_exp)
```

6.408 TrypticIterator

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/TrypticIterator.h>" namespace "OpenMS":

  cdef cppclass TrypticIterator(PepIterator) :
    # wrap-inherits:
    #  PepIterator

    TrypticIterator()
    TrypticIterator(TrypticIterator)
    bool isDigestingEnd(char aa1, char aa2)
    String getProductName()
```

6.409 TwoDOptimization

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/TRANSFORMATIONS/RAW2PEAK/TwoDOptimization.h>" namespace\
       "OpenMS":

  cdef cppclass TwoDOptimization(DefaultParamHandler) :
    # wrap-inherits:
    #  DefaultParamHandler
    TwoDOptimization()
    TwoDOptimization(TwoDOptimization)
    double getMZTolerance()
    void setMZTolerance(double tolerance_mz)
    double getMaxPeakDistance()
    void setMaxPeakDistance(double max_peak_distance)
    UInt getMaxIterations()
    void setMaxIterations(UInt max_iteration)
    # NAMESPACE # OptimizationFunctions::PenaltyFactorsIntensity  getPenalties()
    # NAMESPACE # void setPenalties(OptimizationFunctions::PenaltyFactorsIntensity &\
       penalties)
    # TEMPLATE # void optimize(InputSpectrumIterator first, InputSpectrumIterator\
       last, MSExperiment[ OutputPeakType ] & ms_exp, bool real2D)
```

6.410 Types

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/config.h>":
  pass

cdef extern from "<OpenMS/CONCEPT/Types.h>" namespace "OpenMS":

  ctypedef signed int Int32   "OPENMS_INT32_TYPE"
  ctypedef signed long Int64  "OPENMS_INT64_TYPE"
  ctypedef unsigned int UInt32  "OPENMS_UINT32_TYPE"
  ctypedef unsigned long UInt64 "OPENMS_UINT64_TYPE"
  ctypedef time_t   Time
  ctypedef unsigned int UInt
  ctypedef int    Int
  ctypedef unsigned long   UID "OPENMS_UINT64_TYPE"
  ctypedef size_t  Size
  ctypedef ptrdiff_t SignedSize
```

6.411 UnimodXMLFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/UnimodXMLFile.h>" namespace "OpenMS":

  cdef cppclass UnimodXMLFile(XMLFile) :
    # wrap-inherits:
    #  XMLFile

    UnimodXMLFile()
    UnimodXMLFile(UnimodXMLFile)  #wrap-ignore
```

6.412 UniqueIdGenerator

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CONCEPT/UniqueIdGenerator.h>" namespace "OpenMS":

  cdef cppclass UniqueIdGenerator:

    UInt64 getUniqueId()

    void setSeed(UInt64)

    UInt64 getSeed()
```

6.413 UniqueIdInterface

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CONCEPT/UniqueIdInterface.h>" namespace "OpenMS":

  cdef cppclass UniqueIdInterface:
    # wrap-ignore

    UniqueIdInterface()
    UniqueIdInterface(UniqueIdInterface)

    Size getUniqueId()
    Size clearUniqueId()
    void swap(UniqueIdInterface)  # wrap-ignore
    Size hasValidUniqueId()
    Size hasInvalidUniqueId()
    void setUniqueId(UInt64 rhs)
    Size ensureUniqueId()

    bool isValid(UInt64 unique_id)

cdef extern from "<OpenMS/CONCEPT/UniqueIdInterface.h>" namespace\
       "OpenMS::UniqueIdInterface":

    Size setUniqueId() # wrap-ignore
```

6.414 VersionInfo

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CONCEPT/VersionInfo.h>" namespace "OpenMS":

  cdef cppclass VersionInfo:
    pass

cdef extern from "<OpenMS/CONCEPT/VersionInfo.h>" namespace "OpenMS::VersionInfo":

  cdef cppclass VersionDetails:
    Int version_major
    Int version_minor
    Int version_patch
    VersionDetails()
    VersionDetails(VersionDetails)    #wrap-ignore
```

```
    bool operator<(VersionDetails)
    bool operator==(VersionDetails)
    bool operator>(VersionDetails)

  VersionDetails getVersionStruct()   #wrap-attach:VersionInfo
  String getVersion()    #wrap-attach:VersionInfo
  String getTime()     #wrap-attach:VersionInfo
  String getRevision()   #wrap-attach:VersionInfo
  String getBranch()     #wrap-attach:VersionInfo

cdef extern from "<OpenMS/CONCEPT/VersionInfo.h>" namespace\
       "OpenMS::VersionInfo::VersionDetails":

  VersionDetails create(String) #wrap-attach:VersionDetails
```

6.415 Weights

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/CHEMISTRY/MASSDECOMPOSITION/IMS/Weights.h>" namespace\
       "OpenMS::ims::Weights":

  ctypedef double alphabet_mass_type
  ctypedef long unsigned int weight_type
  ctypedef libcpp_vector[weight_type] weights_type

cdef extern from "<OpenMS/CHEMISTRY/MASSDECOMPOSITION/IMS/Weights.h>" namespace\
       "OpenMS::ims":

  cdef cppclass IMSWeights "OpenMS::ims::Weights":
    IMSWeights()
    IMSWeights(IMSWeights)
    size_type size()
    weight_type getWeight(size_type i)
    void setPrecision(alphabet_mass_type precision)
    alphabet_mass_type getPrecision()
    weight_type back()
    alphabet_mass_type getAlphabetMass(size_type i)
    alphabet_mass_type getParentMass(libcpp_vector[ unsigned int ] & decomposition)
    void swap(size_type index1, size_type index2)
    bool divideByGCD()
    alphabet_mass_type getMinRoundingError()
    alphabet_mass_type getMaxRoundingError()
```

6.416 WindowMower

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FILTERING/TRANSFORMERS/WindowMower.h>" namespace\
       "OpenMS":

  cdef cppclass WindowMower(DefaultParamHandler):
    # wrap-inherits:
    #  DefaultParamHandler

    WindowMower()
    WindowMower(WindowMower)  #wrap-ignore

    void filterPeakSpectrumForTopNInSlidingWindow(MSSpectrum[Peak1D] & spectrum)
    void filterPeakSpectrumForTopNInJumpingWindow(MSSpectrum[Peak1D] & spectrum)

    void filterPeakSpectrum(MSSpectrum[Peak1D] & spec)
    void filterPeakMap(MSExperiment[Peak1D, ChromatogramPeak] & exp)
```

6.417 XMLFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/XMLFile.h>" namespace "OpenMS::Internal":

  cdef cppclass XMLFile "OpenMS::Internal::XMLFile":
    XMLFile()
    XMLFile(XMLFile)  #wrap-ignore
    XMLFile(String & schema_location, String & version)
    # NAMESPACE # bool isValid(String & filename, std::ostream & os)
    String getVersion()
```

6.418 XMLHandler

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/HANDLERS/XMLHandler.h>" namespace\
       "OpenMS::Internal":

  cdef cppclass XMLHandler:
```

```
    XMLHandler()  #wrap-ignore
    XMLHandler(XMLHandler)  #wrap-ignore
    # NAMESPACE # void fatalError(xercesc::SAXParseException & exception)
    # NAMESPACE # void error(xercesc::SAXParseException & exception)
    # NAMESPACE # void warning(xercesc::SAXParseException & exception)
    XMLHandler(String & filename, String & version)
    void reset()
    # TODO cdash might parse out "fatalError" statements and interpret them
    # as compilation failure...
    # void fatalError(ActionMode mode, String & msg, UInt line, UInt column)
    void error(ActionMode mode, String & msg, UInt line, UInt column)
    void warning(ActionMode mode, String & msg, UInt line, UInt column)
    # POINTER # void characters(XMLCh *chars, XMLSize_t length)
    # NAMESPACE # # POINTER # void startElement(XMLCh *uri, XMLCh *localname, XMLCh\
       *qname, xercesc::Attributes & attrs)
    # POINTER # void endElement(XMLCh *uri, XMLCh *localname, XMLCh *qname)
    # NAMESPACE # void writeTo(std::ostream & )
    String errorString()

cdef extern from "<OpenMS/FORMAT/HANDLERS/XMLHandler.h>" namespace\
       "OpenMS::Internal::XMLHandler":
  cdef enum ActionMode "OpenMS::Internal::XMLHandler::ActionMode":
    #wrap-attach:
    #  XMLHandler
    LOAD
    STORE
```

6.419 XTandemInfile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/XTandemInfile.h>" namespace "OpenMS":

  cdef cppclass XTandemInfile: # (XMLFile):
    XTandemInfile()
    XTandemInfile(XTandemInfile)  #wrap-ignore
    void setFragmentMassTolerance(double tolerance)
    double getFragmentMassTolerance()
    void setPrecursorMassTolerancePlus(double tol)
    double getPrecursorMassTolerancePlus()
    void setPrecursorMassToleranceMinus(double tol)
    double getPrecursorMassToleranceMinus()
    void setPrecursorErrorType(MassType mono_isotopic)
    MassType getPrecursorErrorType()
    void setFragmentMassErrorUnit(ErrorUnit unit)
```

```
    ErrorUnit getFragmentMassErrorUnit()
    void setPrecursorMassErrorUnit(ErrorUnit unit)
    ErrorUnit getPrecursorMassErrorUnit()
    void setNumberOfThreads(UInt threads)
    UInt getNumberOfThreads()
    void setModifications(ModificationDefinitionsSet & mods)
    ModificationDefinitionsSet  getModifications()
    void setOutputFilename(String & output)
    String  getOutputFilename()
    void setInputFilename(String & input_file)
    String  getInputFilename()
    void setTaxonomyFilename(String & filename)
    String  getTaxonomyFilename()
    void setDefaultParametersFilename(String & filename)
    String  getDefaultParametersFilename()
    void setTaxon(String & taxon)
    String  getTaxon()
    void setMaxPrecursorCharge(Int max_charge)
    Int getMaxPrecursorCharge()
    void setNumberOfMissedCleavages(UInt missed_cleavages)
    UInt getNumberOfMissedCleavages()
    void setOutputResults(String result)
    String getOutputResults()
    void setMaxValidEValue(double value)
    double getMaxValidEValue()
    bool isRefining()
    void setSemiCleavage(bool semi_cleavage)
    void setAllowIsotopeError(bool allow_isotope_error)
    void setRefine(bool refine)
    void write(String & filename)
    void load(String & filename)
    void setCleavageSite(String cleavage_site)
    String getCleavageSite()

cdef extern from "<OpenMS/FORMAT/XTandemInfile.h>" namespace\
       "OpenMS::XTandemInfile":
  cdef enum ErrorUnit "OpenMS::XTandemInfile::ErrorUnit":
    #wrap-attach:
    #  XTandemInfile
    DALTONS
    PPM

cdef extern from "<OpenMS/FORMAT/XTandemInfile.h>" namespace\
       "OpenMS::XTandemInfile":
  cdef enum MassType "OpenMS::XTandemInfile::MassType":
    #wrap-attach:
    #  XTandemInfile
    MONOISOTOPIC
```

```
    AVERAGE
```

6.420 XTandemXMLFile

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<OpenMS/FORMAT/XTandemXMLFile.h>" namespace "OpenMS":

  cdef cppclass XTandemXMLFile:

    XTandemXMLFile()

    void load(String filename, ProteinIdentification & protein_identification,\
       libcpp_vector[PeptideIdentification] & id_data)

    void setModificationDefinitionsSet(ModificationDefinitionsSet rhs)
```

6.421 streampos

→ *Link to OpenMS documentation*

Wrapped functions in Python:

```
cdef extern from "<iostream>" namespace "std":
  cdef cppclass streampos:
    streampos()
    streampos(streampos &)
```