



## [museg] SDK (FeatureSimilarity) 2.6.4

by zplane.development  
(c) 2016 zplane.development GmbH & Co. KG

June 14, 2016

# Contents

<b>1 [museg] SDK FeatureSimilarity Documentation</b>	<b>2</b>
1.1 Introduction . . . . .	2
1.2 Input Data . . . . .	2
1.3 Internal API Documentation . . . . .	2
1.3.1 Naming Conventions . . . . .	2
1.3.2 Memory Allocation . . . . .	3
1.3.3 Instance Handling Functions . . . . .	3
1.3.4 Initialization Functions . . . . .	3
1.3.5 Processing Functions . . . . .	3
1.3.6 Additional Functions . . . . .	4
1.3.7 Calling Conventions . . . . .	4
1.4 Support . . . . .	5
<b>2 Directory Hierarchy</b>	<b>6</b>
2.1 Directories . . . . .	6
<b>3 Data Structure Index</b>	<b>6</b>
3.1 Data Structures . . . . .	6
<b>4 File Index</b>	<b>6</b>
4.1 File List . . . . .	6
<b>5 Directory Documentation</b>	<b>7</b>
5.1 E:/Visual Studio Projects/zplane/CMakeTestDir/museg/incl/ Directory Reference . . . . .	7
<b>6 Data Structure Documentation</b>	<b>7</b>
6.1 ClassificationResult_t_tag Struct Reference . . . . .	7
6.1.1 Detailed Description . . . . .	8
6.1.2 Field Documentation . . . . .	8
6.2 CSegmentation Class Reference . . . . .	8
6.2.1 Detailed Description . . . . .	11
6.2.2 Member Typedef Documentation . . . . .	11
6.2.3 Member Enumeration Documentation . . . . .	12
6.2.4 Constructor & Destructor Documentation . . . . .	14
6.2.5 Member Function Documentation . . . . .	14
6.2.6 Field Documentation . . . . .	16
6.3 CTraining Class Reference . . . . .	20
6.3.1 Detailed Description . . . . .	20
6.3.2 Constructor & Destructor Documentation . . . . .	21
6.3.3 Member Function Documentation . . . . .	21
6.3.4 Field Documentation . . . . .	21
6.4 FeatureSimilarityResult_t_tag Struct Reference . . . . .	22
6.4.1 Detailed Description . . . . .	22
6.4.2 Field Documentation . . . . .	22
6.5 SegmentationResult_t_tag Struct Reference . . . . .	22
6.5.1 Detailed Description . . . . .	23
6.5.2 Field Documentation . . . . .	23

6.6	TrainingSetInfo_t_tag Struct Reference . . . . .	23
6.6.1	Detailed Description . . . . .	24
6.6.2	Field Documentation . . . . .	24
<b>7</b>	<b>File Documentation</b>	<b>24</b>
7.1	Classification_C.h File Reference . . . . .	24
7.1.1	Detailed Description . . . . .	25
7.1.2	Typedef Documentation . . . . .	25
7.1.3	Function Documentation . . . . .	26
7.2	FeatureExtraction_C.h File Reference . . . . .	31
7.2.1	Detailed Description . . . . .	32
7.2.2	Function Documentation . . . . .	32
7.3	featuresimilarity.txt File Reference . . . . .	39
7.4	FeatureSimilarity_C.h File Reference . . . . .	39
7.4.1	Detailed Description . . . . .	40
7.4.2	Typedef Documentation . . . . .	40
7.4.3	Function Documentation . . . . .	40
7.5	Globals.h File Reference . . . . .	42
7.5.1	Detailed Description . . . . .	43
7.5.2	Define Documentation . . . . .	43
7.5.3	Typedef Documentation . . . . .	43
7.5.4	Enumeration Type Documentation . . . . .	43
7.6	Segmentation.h File Reference . . . . .	46
7.6.1	Detailed Description . . . . .	46
7.6.2	Define Documentation . . . . .	46
7.7	Segmentation_C.h File Reference . . . . .	47
7.7.1	Typedef Documentation . . . . .	48
7.7.2	Enumeration Type Documentation . . . . .	48
7.7.3	Function Documentation . . . . .	49
7.8	Training.h File Reference . . . . .	54
7.8.1	Detailed Description . . . . .	54
7.9	Training_C.h File Reference . . . . .	54
7.9.1	Detailed Description . . . . .	55
7.9.2	Typedef Documentation . . . . .	55
7.9.3	Function Documentation . . . . .	55

# 1 [museg] SDK FeatureSimilarity Documentation

## 1.1 Introduction

The FeatureSimilarity part of the [museg] SDK allows to compute a similarity measure between different series of features as extracted by the FeatureExtraction. More specifically, it computes the similarity between two feature sequences of different length. Algorithmically speaking, the process detects the maximum similarity between the shifted sequences.

There is no specific to train the SDK except for feature mean values which are explained below.

## 1.2 Input Data

The FeatureSimilarity API expects the following input data:

- Matrix 1 (Reference): Feature Data (Dimensions: Feature x Observation) as extracted in the Feature Extraction
- Matrix 2 (Test): Feature Data including the same number of features but less or equal number of observations as matrix 1
- Feature Mean Values: these have been previously extracted from a large data set. The feature mean values are a vector of the length of dimension 1 of both matrices given above. Each entry is the arithmetic mean or (better) the median value of a single feature computed over a larget set of data.

## 1.3 Internal API Documentation

The SDK's library provides a C-API which is available in the file [FeatureSimilarity\\_C.h](#). All required variable types are either defined in this file, in the additional include file [Globals.h](#) or are standard C-types.

### 1.3.1 Naming Conventions

When talking about **frames**, the number of audio samples per channel is meant. I.e. 512 stereo frames correspond to 1024 float values (samples). If the sample size is 32bit float, one sample has a memory usage of 4 byte.

The term **feature** refers to a kind of meta information that is extracted from the audio data. In this context, several features are extracted, where every feature consists of a floating point value for each time frame.

If a two-dimensional buffer or array (e.g. `afArray[i][j]`) is easier to be interpreted as matrix, the first dimension (`i`) will be referred to as row and the second dimension (`j`) as column.

### 1.3.2 Memory Allocation

The SDK does not allocate buffers handled by the calling application. The input and output buffers have to be allocated by the calling application. Audio data buffers are allocated in interleaved format as single arrays of length [frames].

### 1.3.3 Instance Handling Functions

- **FeatSim.CreateInstance** (`void** pphFeatureSimilarityHandle`)

Creates a new instance for FeatureSimilarity. The parameter `pphFeatureSimilarityHandle` is written.

Note that the call of this function implies the subsequent call of [FeatSim\\_Initialize](#).

The function returns 0 in the case of no error.

- **FeatSim\_DestroyInstance** (`void** pphFeatureSimilarityHandle`)

Destroys an instance of FeatureSimilarity. The parameter `pphFeatureSimilarityHandle` is set to NULL.

The function returns 0 in the case of no error.

### 1.3.4 Initialization Functions

- **FeatSim\_Initialize** (`void* phFeatureSimilarityHandle, float *pfMeans, int aiDim1[2], int aiDim2[2]`)

Allocates memory and initializes the internal variables.. The parameter `phFeatureSimilarityHandle` is the handle to the previously created instance and the parameter `pfMeans` contains the feature means as described above; the length of the buffer `pfMeans` is `aiDim1[0]` and `aiDim2[0]`. The parameters `aiDim1[1]` and `aiDim2[1]` contain the number of observations per feature for input sequence 1 and 2, respectively.

The function returns 0 in the case of no error.

### 1.3.5 Processing Functions

- **FeatSim\_Process** (`void* phFeatureSimilarityHandle, float **ppfFeatureMatrix1, float **ppfFeatureMatrix2`)

Processes the generation of the output data. This function produces the highest workload of all API functions. The parameter `phFeatureSimilarityHandle` is the handle to the previously created instance. The function [FeatSim\\_Process](#) requires feature data as given in the parameters `ppfFeatureMatrix1` and `ppfFeatureMatrix2`. The dimensions of these matrices have to correspond to the parameters `aiDim1` an `aiDim2` as given in [FeatSim\\_Initialize](#).

The function returns 0 in the case of no error.

- **FeatSim\_GetResult** (`void* phFeatureSimilarityHandle, FeatureSimilarityResult_t* pstResult`)

Copies the calculated result to the structure `pstResult`. The parameter `phFeatureSimilarityHandle` is the handle to the previously created instance. Note that the memory of `pstResult` points to has to be allocated by the user of the SDK. - The content of `pstResult` is two values, `fMaxLikelihood` in the range 0...1 and `i-FeatureMatrixIndex` which indicates the index of where the maximum likelihood has been found.

The function returns 0 in the case of no error.

### 1.3.6 Additional Functions

- **FeatSim\_GetBuildDateString** ()

Returns a char containing the build date of the training library. This function may also be called before instance creation.

- **FeatSim\_GetVersion** (`Version_t eVersionIdx`)

Returns an int with the (major, minor,...) version of the training library. This function may also be called before instance creation.

### 1.3.7 Calling Conventions

This is a step-by-step introduction for the usage of the SDK. The complete code can be found in the example source file FeatureSimilarityCIMain.cpp.

In the first step, a handle to the instance has to be declared:

```
void *phFeatureSimilarity = 0; //  
instance handle
```

This command line example expects text files for the input data. After we allocated the necessary memory we can read the feature data as well as the feature means from the files

```
// alloc input feature matrices  
for (int k = 0; k < kNumInputs; k++)  
{  
    // rows: different features  
    appfFeatureMatrix[k] = new float *[aaifeatureMatrixSize[0][0]];  
    // cols: observations  
    aaiFeatureMatrixSize[k][1] = aaIndices[k][kStop]-aaIndices[k][kStart]  
    ] + 1;  
    for (int i = 0; i < aaiFeatureMatrixSize[0][0]; i++)  
        appfFeatureMatrix[k][i] = new float [aaifeatureMatrixSize[k][1]]  
    };  
  
    // read matrices from file  
    for (int k = 0; k < kNumInputs; k++)  
        CLReadMatrixFromFile (appfFeatureMatrix[k], FInputFile[k], aaIndices[k]  
        ], piFeatureIndices, aaiFeatureMatrixSize[0][0]);  
    if (*pcMeansPath)
```

```
{
    // if there is a means file (i.e. path not empty) read the feature
    means...
    int iRow      = 0;
    int aiMeans[2];
    aiMeans[0]    = 0;
    aiMeans[1]    = aaiFeatureMatrixSize[0][0]-1;
    CLReadMatrixFromFile (&pfMeans, FMeans, aiMeans, &iRow, 1);
}
```

Then, the instance is created and initialized

```
// create class instances and initialize the feature extraction
FeatSim_CreateInstance ( &phFeatureSimilarity);

//initialize the class
FeatSim_Initialize ( phFeatureSimilarity,
    pfMeans,
    aaiFeatureMatrixSize[0],
    aaiFeatureMatrixSize[1]);
```

After that, we can calculate the similarity between the two sequences:

```
FeatSim_Process (phFeatureSimilarity,
    appfFeatureMatrix[kInputLong],
    appfFeatureMatrix[kInputShort]);
```

Finally, we can retrieve the result and print it

```
// get result
FeatSim_GetResult (phFeatureSimilarity,
    &stSimilarityResult);

// plot and print result
std::cout << "Result: " << stSimilarityResult.fMaxLikelihood << "\t" <<
    stSimilarityResult.iFeatureMatrixIndex << endl;
```

In the end, the instance can be destroyed by

```
// destroy instances
FeatSim_DestroyInstance (phFeatureSimilarity);
```

## 1.4 Support

Support for the source code is - within the limits of the agreement - available from:

**zplane.development**  
 grunewaldstr. 83  
 d-10823 berlin  
 germany  
 fon: +49.30.854 09 15.0

fax: +49.30.854 09 15.5

@: [info@zplane.de](mailto:info@zplane.de)

## 2 Directory Hierarchy

### 2.1 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

<b>incl</b>	<b>7</b>
-------------	----------

## 3 Data Structure Index

### 3.1 Data Structures

Here are the data structures with brief descriptions:

<b>ClassificationResult_t_tag</b>	<b>7</b>
<b>CSegmentation</b>	<b>8</b>
<b>CTraining</b>	<b>20</b>
<b>FeatureSimilarityResult_t_tag</b>	<b>22</b>
<b>SegmentationResult_t_tag</b>	<b>22</b>
<b>TrainingSetInfo_t_tag</b>	<b>23</b>
<b>Info structure on the training data</b>	

## 4 File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

<b>Classification_C.h</b> C-wrapper for the CClassification class	<b>24</b>
<b>FeatureExtraction_C.h</b> C-wrapper for the CFeatureExtraction class	<b>31</b>
<b>FeatureSimilarity_C.h</b> C-wrapper for the CFeatureSimilarity class	<b>39</b>

<b>Globals.h</b>		
Some global constants/types		<b>42</b>
<b>Segmentation.h</b>		
Interface of the <b>CSegmentation</b> class		<b>46</b>
<b>Segmentation_C.h</b>		
		<b>47</b>
<b>Training.h</b>		
Interface of the <b>CTraining</b> class		<b>54</b>
<b>Training_C.h</b>		
C-interface wrapper for the Training		<b>54</b>

## 5 Directory Documentation

### 5.1 E:/Visual Studio Projects/zplane/CMakeTestDir/museg/incl/ Directory Reference

Directory dependency graph for E:/Visual Studio Projects/zplane/CMakeTestDir/museg/incl/:  
:

#### Files

- file **Classification\_C.h**  
*C-wrapper for the CClassification class.*
- file **FeatureExtraction\_C.h**  
*C-wrapper for the CFeatureExtraction class.*
- file **FeatureSimilarity\_C.h**  
*C-wrapper for the CFeatureSimilarity class.*
- file **Globals.h**  
*some global constants/types.*
- file **Segmentation.h**  
*interface of the CSegmentation class.*
- file **Segmentation\_C.h**
- file **Training.h**  
*interface of the CTraining class.*
- file **Training\_C.h**  
*C-interface wrapper for the Training.*

## 6 Data Structure Documentation

### 6.1 ClassificationResult\_t\_tag Struct Reference

```
#include <Classification_C.h>
```

### Data Fields

- float **iStartTimeInS**  
*segment start*
- float **iStopTimeInS**  
*segment stop*
- int **iEstimatedClassIdx**  
*class*
- float **fEstimationReliability**  
*estimate of result reliability*

#### 6.1.1 Detailed Description

declaration of result structure

Definition at line 55 of file Classification\_C.h.

#### 6.1.2 Field Documentation

##### 6.1.2.1 float ClassificationResult\_t\_tag::fEstimationReliability

estimate of result reliability

Definition at line 62 of file Classification\_C.h.

##### 6.1.2.2 int ClassificationResult\_t\_tag::iEstimatedClassIdx

class

Definition at line 60 of file Classification\_C.h.

##### 6.1.2.3 float ClassificationResult\_t\_tag::iStartTimeInS

segment start

Definition at line 57 of file Classification\_C.h.

##### 6.1.2.4 float ClassificationResult\_t\_tag::iStopTimeInS

segment stop

Definition at line 57 of file Classification\_C.h.

The documentation for this struct was generated from the following file:

- [Classification\\_C.h](#)

## 6.2 CSegmentation Class Reference

```
#include <Segmentation.h>
```

Collaboration diagram for CSegmentation:

### Public Member Functions

- `CSegmentation` (int iSampleRate, int iNumberOfChannels)
- virtual ~`CSegmentation` ()
- zERROR `Initialize` (int iOverallInputFileLengthInFrames)
- zERROR `PreProcess` (float \*pfInputBufferInterleaved, int iNumberOfFrames)
- zERROR `Process` (float \*pfInputBufferInterleaved, int iNumberOfFrames)
- zERROR `FinishProcess` ()
- zERROR `SetTxtFilePath` (char \*pcTxtFilePath)
- zERROR `PostProcess` (float fTransitionWeight, float fMinimumClassTimeInS, float fAPrioriProbabilityOfMusic, float fMinimumEnergy, float \*\*ppfRawResult=0)
- zERROR `GetSizeOfIntermediateResult` (zINT \*piRows, zINT \*piColumns)
- zERROR `GetIntermediateResult` (zFLOAT \*\*pfIntermediateResult)
- zERROR `SetIntermediateResult` (zFLOAT \*\*pfIntermediateResult, zINT iRows, zINT iColumns)
- int `GetSizeOfResult` ()
- zERROR `GetResult` (`SegmentationResult_t` \*pstResult)

### Static Public Member Functions

- static zERROR `CreateInstance` (`CSegmentation` \*&pCSegmentation, int iSampleRate, int iNumberOfChannels)
- static zERROR `DestroyInstance` (`CSEGMENTATION` \*&pCSegmentation)
- static const int `GetVersion` (const `Version_t` eVersionIdx)
- static const char \* `GetBuildDate` ()

### Private Types

- enum `ProcessBuffers_t_tag` { k1, k2, k3, kNumProcessBuffers }
- enum `FFTSizes_t_tag` { kShort, kLong, kNumFFTSizes }
- enum `Indices_t_tag` { kStart, kStop, kNumIndices }
- enum `Phases_t_tag` { kPrevious, kCurrent, kNumPhaseStates }
- enum `Features2Extract_t_tag` { kLoudness = 0, kPeakSteadiness, kNoiseness, kNoiseness2, kNoiseness3, kNoiseness4, kRhythmicness, kCentroid, kSpread, kMidBandFlatness, kRollOff, kFlux, kMonoStrength, kMFCC, kNumFeatures = kMFCC + kNumMelCoeffs }
- enum `SubFeatures_t_tag` { kMean = 0, kStd, kDerivStd, kMax2Mean, kMaxRegularity, kNumSubFeatures }
- typedef enum `CSEGMENTATION::ProcessBuffers_t_tag` `ProcessBuffers_t`
- typedef enum `CSEGMENTATION::FFTSizes_t_tag` `FftSizes_t`
- typedef enum `CSEGMENTATION::Indices_t_tag` `Indices_t`
- typedef enum `CSEGMENTATION::Phases_t_tag` `Phases_t`
- typedef enum `CSEGMENTATION::Features2Extract_t_tag` `Features2Extract_t`
- typedef enum `CSEGMENTATION::SubFeatures_t_tag` `SubFeatures_t`

**Private Member Functions**

- zERROR [CalcFreqTableMpeg7\(\)](#)
- zERROR [CalcFlatnessFreqs\(\)](#)
- zERROR [CalcMFCCFilters\(zINT iNumOfBands, zFLOAT fMinFreq, zFLOAT fMaxFreq\)](#)
- zERROR [CalcMFCC\(zFLOAT \\*pfResult\)](#)
- zFLOAT [CalcMonoStrength\(\)](#)
- zERROR [LoadMatrixFromFile\(zFLOAT \\*\\*ppfMatrix, std::string acFilePath\)](#)
- zERROR [CalcDTW\(zFLOAT \\*\\*ppfSimilarityMatrix, zFLOAT \\*\\*ppfCostMatrix, zINT \\*piPathIdx, zFLOAT fTransitionCost, zINT iNumOfRows, zINT iNumOfColumns\)](#)
- zERROR [RemoveLowEnergyFrames\(zINT \\*piPath, zFLOAT \\*pfLoudness, zINT iLengthOPath, zFLOAT fLoudnessThreshold\)](#)
- zERROR [ScaleLowEnergyFrames\(zFLOAT \\*\\*ppfDistances, zFLOAT \\*pfLoudness, zINT iLengthOfPath\)](#)
- zVOID [CalcPitchSteadiness\(zFLOAT \\*pfAudio, zFLOAT pfResult\[3\]\)](#)
- zVOID [CalcPitchBounds\(\)](#)
- zVOID [CalcPitchSteadinessIndices\(\)](#)
- zINT [PeakPicking\(zFLOAT \\*pfMagSpectrum, zFLOAT \\*pfInstFreq, FftSizes\\_t eFftSize\)](#)
- zFLOAT [RelativeNoisePower\(zFLOAT \\*pfMagSpectrum, zFLOAT \\*pfInstFreq, zFLOAT \\*pfPeakSpectrum, FftSizes\\_t eFftSize\)](#)
- zFLOAT [AbsoluteNoisePower\(zFLOAT \\*pfMagSpectrum, zFLOAT \\*pfInstFreq, zFLOAT \\*pfPeakSpectrum, FftSizes\\_t eFftSize\)](#)
- zFLOAT [CalcSpecSlope\(zFLOAT \\*pfMagSpec, zINT iLength\)](#)
- zVOID [CalcRhythmFeature\(\)](#)
- zFLOAT [CalcBackgroundNoisePower\(zFLOAT \\*pfComplexSpectrum, FftSizes\\_t eFFTSize\)](#)

**Private Attributes**

- zFLOAT [m\\_fSampleRate](#)
- zFLOAT [m\\_f.MaxValue](#)
- zFLOAT [m\\_fStartTime](#)
- zINT [m\\_iNumberOfChannels](#)
- zINT [m\\_aiProcessBlockSize \[kNumFFTSizes\]](#)
- zINT [m\\_iProcessHopSizeInFrames](#)
- zINT [m\\_aiCurrentTimeIndex \[kNumFFTSizes\]](#)
- zINT [m\\_iFileLengthInFrames](#)
- zINT [m\\_iMpeg7StartIdx](#)
- zINT [m\\_iNumOfFeatureVectors](#)
- zINT [m\\_iCompleteNumOfFeatures](#)
- zINT [m\\_iTextureWindowLength](#)
- zINT [m\\_iTextureWindowHop](#)
- zINT [m\\_iRhythmWindowLength](#)
- zINT [m\\_aiFlatnessBounds \[4\]](#)

- zINT [m\\_iLengthOfResult](#)
- zFLOAT32 \*\* [m\\_ppfFeatures](#)
- zFLOAT32 \* [m\\_apfProcessBuffer](#) [kNumProcessBuffers]
- zFLOAT32 \* [m\\_apfPrevFFT](#) [kNumFFTSizes]
- zFLOAT32 \* [m\\_pfLogFreqs](#)
- zFLOAT32 \* [m\\_pfPreEmphasis](#)
- zFLOAT32 \*\* [m\\_ppfMFCCFilters](#)
- zFLOAT32 \*\* [m\\_ppfDCTCoeffs](#)
- zFLOAT32 \* [m\\_pfMFCCBuffer](#)
- CRingBuffer< zFLOAT32 > \* [m\\_pCRingBuffer](#)
- CRingBuffer< zFLOAT32 > \* [m\\_apCFeatureBuffer](#) [kNumFeatures]
- CRingBuffer< zFLOAT32 > \* [m\\_pOnsetCurveBuffer](#)
- CzplfFFT\_If \* [m\\_apFFTInstance](#) [kNumFFTSizes]
- [SegmentationResult\\_t](#) \* [m\\_pstResult](#)
- std::string [m\\_strTxtFilePath](#)
- CParametricEqIf \* [m\\_apfPreFilters](#) [kNumPreFilters]
- zFLOAT [m\\_fLowPassCoeff](#)
- zFLOAT \* [m\\_pfPeakSteadiness](#)
- zFLOAT \* [m\\_apfPhase](#) [kNumPhaseStates]
- zFLOAT \* [m\\_pfOmega](#)
- zINT \* [m\\_apiPitchBounds](#) [kNumIndices]
- zINT [m\\_aaiIndices](#) [kNumFFTSizes][kNumIndices]
- zINT [m\\_iNumOfPitches](#)

### 6.2.1 Detailed Description

Definition at line 158 of file Segmentation.h.

### 6.2.2 Member Typedef Documentation

- 6.2.2.1 **typedef enum CSegmentation::Features2Extract\_t\_tag**  
[CSegmentation::Features2Extract\\_t](#) [private]
- 6.2.2.2 **typedef enum CSegmentation::FFTSizes\_t\_tag CSegmentation::FftSizes\_t**  
[private]
- 6.2.2.3 **typedef enum CSegmentation::Indices\_t\_tag CSegmentation::Indices\_t**  
[private]
- 6.2.2.4 **typedef enum CSegmentation::Phases\_t\_tag CSegmentation::Phases\_t**  
[private]
- 6.2.2.5 **typedef enum CSegmentation::ProcessBuffers\_t\_tag**  
[CSegmentation::ProcessBuffers\\_t](#) [private]
- 6.2.2.6 **typedef enum CSegmentation::SubFeatures\_t\_tag**  
[CSegmentation::SubFeatures\\_t](#) [private]

### 6.2.3 Member Enumeration Documentation

#### 6.2.3.1 enum CSegmentation::Features2Extract\_t\_tag [private]

Enumerator:

- kLoudbess*
- kPeakSteadiness*
- kNoiseness*
- kNoiseness2*
- kNoiseness3*
- kNoiseness4*
- kRhythmicness*
- kCentroid*
- kSpread*
- kMidBandFlatness*
- kRollOff*
- kFlux*
- kMonoStrength*
- kMFCC*
- kNumFeatures*

Definition at line 251 of file Segmentation.h.

```
{
    kLoudbess      = 0,
#ifndef V15ONLY
//      kStereoCorr,
//      kStereoMSLevel,
#endif
        kPeakSteadiness,
#endif CLIP_DETECTION
        kClipping,
#endif
        kNoiseness,
        kNoiseness2,
        kNoiseness3,
        kNoiseness4,
#endif RHYTHM
        kRhythmicness,
#endif
        kCentroid,
        kSpread,
        kMidBandFlatness,
        kRollOff,
        kFlux,
        kMonoStrength,
        kMFCC,
        kNumFeatures     = kMFCC + kNumMelCoeffs
    //kNumFeatures
} Features2Extract_t;
```

**6.2.3.2 enum CSegmentation::FFTSizes\_t\_tag [private]****Enumerator:**

*kShort*  
*kLong*  
*kNumFFTSizes*

Definition at line 206 of file Segmentation.h.

```
{  
    kShort,  
    kLong,  
  
    kNumFFTSizes,  
} FftSizes_t;
```

**6.2.3.3 enum CSegmentation::Indices\_t\_tag [private]****Enumerator:**

*kStart*  
*kStop*  
*kNumIndices*

Definition at line 214 of file Segmentation.h.

```
{  
    kStart,  
    kStop,  
  
    kNumIndices  
} Indices_t;
```

**6.2.3.4 enum CSegmentation::Phases\_t\_tag [private]****Enumerator:**

*kPrevious*  
*kCurrent*  
*kNumPhaseStates*

Definition at line 222 of file Segmentation.h.

```
{  
    kPrevious,  
    kCurrent,  
  
    kNumPhaseStates  
} Phases_t;
```

**6.2.3.5 enum CSegmentation::ProcessBuffers\_t\_tag [private]****Enumerator:**

*k1*  
*k2*  
*k3*  
*kNumProcessBuffers*

Definition at line 197 of file Segmentation.h.

```
{  
    k1,  
    k2,  
    k3,  
  
    kNumProcessBuffers  
} ProcessBuffers_t;
```

**6.2.3.6 enum CSegmentation::SubFeatures\_t\_tag [private]****Enumerator:**

*kMean*  
*kStd*  
*kDerivStd*  
*kMax2Mean*  
*kMaxRegularity*  
*kNumSubFeatures*

Definition at line 281 of file Segmentation.h.

```
{  
    kMean    = 0,  
    kStd,  
    kDerivStd,  
    kMax2Mean,  
    kMaxRegularity,  
  
    kNumSubFeatures  
} SubFeatures_t;
```

**6.2.4 Constructor & Destructor Documentation****6.2.4.1 CSegmentation::CSegmentation ( int *iSampleRate*, int *iNumberOfChannels* )****6.2.4.2 virtual CSegmentation::~CSegmentation( ) [virtual]****6.2.5 Member Function Documentation**

- 6.2.5.1 **zFLOAT** CSegmentation::AbsoluteNoisePower ( **zFLOAT** \* *pfMagSpectrum*,  
**zFLOAT** \* *pfInstFreq*, **zFLOAT** \* *pfPeakSpectrum*, **FftSizes\_t** *eFftSize* )  
[private]
- 6.2.5.2 **zFLOAT** CSegmentation::CalcBackgroundNoisePower ( **zFLOAT** \*  
*pfComplexSpectrum*, **FftSizes\_t** *eFFTSize* ) [private]
- 6.2.5.3 **zERROR** CSegmentation::CalcDTW ( **zFLOAT** \*\* *ppfSimilarityMatrix*,  
**zFLOAT** \*\* *ppfCostMatrix*, **zINT** \* *piPathIdx*, **zFLOAT** *fTransitionCost*, **zINT**  
*iNumOfRows*, **zINT** *iNumOfColumns* ) [private]
- 6.2.5.4 **zERROR** CSegmentation::CalcFlatnessFreqs ( ) [private]
- 6.2.5.5 **zERROR** CSegmentation::CalcFreqTableMpeg7 ( ) [private]
- 6.2.5.6 **zERROR** CSegmentation::CalcMFCC ( **zFLOAT** \* *pfResult* ) [private]
- 6.2.5.7 **zERROR** CSegmentation::CalcMFCCFilters ( **zINT** *iNumOfBands*, **zFLOAT**  
*fMinFreq*, **zFLOAT** *fMaxFreq* ) [private]
- 6.2.5.8 **zFLOAT** CSegmentation::CalcMonoStrength ( ) [private]
- 6.2.5.9 **zVOID** CSegmentation::CalcPitchBounds ( ) [private]
- 6.2.5.10 **zVOID** CSegmentation::CalcPitchSteadiness ( **zFLOAT** \* *pfAudio*, **zFLOAT**  
*pfResult[3]* ) [private]
- 6.2.5.11 **zVOID** CSegmentation::CalcPitchSteadinessIndices ( ) [private]
- 6.2.5.12 **zVOID** CSegmentation::CalcRhythmFeature ( ) [private]
- 6.2.5.13 **zFLOAT** CSegmentation::CalcSpecSlope ( **zFLOAT** \* *pfMagSpec*, **zINT**  
*iLength* ) [private]
- 6.2.5.14 static **zERROR** CSegmentation::CreateInstance ( **CSegmentation** \*&  
*pCSegmentation*, **int** *iSampleRate*, **int** *iNumberOfChannels* ) [static]
- 6.2.5.15 static **zERROR** CSegmentation::DestroyInstance ( **CSegmentation** \*&  
*pCSegmentation* ) [static]
- 6.2.5.16 **zERROR** CSegmentation::FinishProcess ( )
- 6.2.5.17 static const **char**\* CSegmentation::GetBuildDate ( ) [static]
- 6.2.5.18 **zERROR** CSegmentation::GetIntermediateResult ( **zFLOAT** \*\*  
*pfIntermediateResult* )
- 6.2.5.19 **zERROR** CSegmentation::GetResult ( **SegmentationResult\_t** \* *pstResult*  
)
- 6.2.5.20 **zERROR** CSegmentation::GetSizeOfIntermediateResult ( **zINT** \* *piRows*,  
**zINT** \* *piColumns* )

- 6.2.5.21 int CSegmentation::GetSizeOfResult ( )
- 6.2.5.22 static const int CSegmentation::GetVersion ( const Version\_t eVersionIdx ) [static]
- 6.2.5.23 zERROR CSegmentation::Initialize ( int iOverallInputFileLengthInFrames )
- 6.2.5.24 zERROR CSegmentation::LoadMatrixFromFile ( zFLOAT \*\* ppfMatrix, std::string acFilePath ) [private]
- 6.2.5.25 zINT CSegmentation::PeakPicking ( zFLOAT \* pfMagSpectrum, zFLOAT \* pfInstFreq, FftSizes\_t eFftSize ) [private]
- 6.2.5.26 zERROR CSegmentation::PostProcess ( float fTransitionWeight, float fMinimumClassTimeInS, float fAPrioriProbabilityOfMusic, float fMinimumEnergy, float \*\* ppfRawResult = 0 )
- 6.2.5.27 zERROR CSegmentation::PreProcess ( float \* pfInputBufferInterleaved, int iNumberOfFrames )
- 6.2.5.28 zERROR CSegmentation::Process ( float \* pfInputBufferInterleaved, int iNumberOfFrames )
- 6.2.5.29 zFLOAT CSegmentation::RelativeNoisePower ( zFLOAT \* pfMagSpectrum, zFLOAT \* pfInstFreq, zFLOAT \* pfPeakSpectrum, FftSizes\_t eFftSize ) [private]
- 6.2.5.30 zERROR CSegmentation::RemoveLowEnergyFrames ( zINT \* piPath, zFLOAT \* pfLoudness, zINT iLengthOPath, zFLOAT fLoudnessThreshold ) [private]
- 6.2.5.31 zERROR CSegmentation::ScaleLowEnergyFrames ( zFLOAT \*\* ppfDistances, zFLOAT \* pfLoudness, zINT iLengthOfPath ) [private]
- 6.2.5.32 zERROR CSegmentation::SetIntermediateResult ( zFLOAT \*\* pfIntermediateResult, zINT iRows, zINT iColumns )
- 6.2.5.33 zERROR CSegmentation::SetTxtFilePath ( char \* pcTxtFilePath )

## 6.2.6 Field Documentation

- 6.2.6.1 zINT CSegmentation::m\_aaiIndices[kNumFFTSizes][kNumIndices] [private]

Definition at line 344 of file Segmentation.h.

- 6.2.6.2 zINT CSegmentation::m\_aiCurrentTimeIndex[kNumFFTSizes] [private]

Definition at line 298 of file Segmentation.h.

**6.2.6.3 zINT CSegmentation::m\_aiFlatnessBounds[4] [private]**

Definition at line 298 of file Segmentation.h.

**6.2.6.4 zINT CSegmentation::m\_aiProcessBlockSize[kNumFFTSizes]  
[private]**

Definition at line 298 of file Segmentation.h.

**6.2.6.5 CRingBuffer<zFLOAT32>\* CSegmentation::m\_apCFeatureBuffer[kNum-  
Features] [private]**

Definition at line 322 of file Segmentation.h.

**6.2.6.6 CzplffFT\_If\* CSegmentation::m\_apFFTInstance[kNumFFTSizes]  
[private]**

Definition at line 326 of file Segmentation.h.

**6.2.6.7 zFLOAT \* CSegmentation::m\_apfPhase[kNumPhaseStates]  
[private]**

Definition at line 340 of file Segmentation.h.

**6.2.6.8 CParametricEqIf\* CSegmentation::m\_apfPreFilters[kNumPreFilters]  
[private]**

Definition at line 333 of file Segmentation.h.

**6.2.6.9 zFLOAT32 \* CSegmentation::m\_apfPrevFFT[kNumFFTSizes]  
[private]**

Definition at line 313 of file Segmentation.h.

**6.2.6.10 zFLOAT32 \* CSegmentation::m\_apfProcessBuffer[kNumProcessBuffers]  
[private]**

Definition at line 313 of file Segmentation.h.

**6.2.6.11 zINT\* CSegmentation::m\_apiPitchBounds[kNumIndices] [private]**

Definition at line 343 of file Segmentation.h.

**6.2.6.12 zFLOAT CSegmentation::m\_fLowPassCoeff [private]**

Definition at line 339 of file Segmentation.h.

**6.2.6.13 zFLOAT CSegmentation::m\_f.MaxValue [private]**

Definition at line 295 of file Segmentation.h.

**6.2.6.14 zFLOAT CSegmentation::m\_fSampleRate [private]**

Definition at line 295 of file Segmentation.h.

**6.2.6.15 zFLOAT CSegmentation::m\_fStartTime [private]**

Definition at line 295 of file Segmentation.h.

**6.2.6.16 zINT CSegmentation::m\_iCompleteNumOfFeatures [private]**

Definition at line 298 of file Segmentation.h.

**6.2.6.17 zINT CSegmentation::m\_iFileLengthInFrames [private]**

Definition at line 298 of file Segmentation.h.

**6.2.6.18 zINT CSegmentation::m\_iLengthOfResult [private]**

Definition at line 298 of file Segmentation.h.

**6.2.6.19 zINT CSegmentation::m\_iMpeg7StartIdx [private]**

Definition at line 298 of file Segmentation.h.

**6.2.6.20 zINT CSegmentation::m\_iNumberOfChannels [private]**

Definition at line 298 of file Segmentation.h.

**6.2.6.21 zINT CSegmentation::m\_iNumOfFeatureVectors [private]**

Definition at line 298 of file Segmentation.h.

**6.2.6.22 zINT CSegmentation::m\_iNumOfPitches [private]**

Definition at line 344 of file Segmentation.h.

**6.2.6.23 zINT CSegmentation::m\_iProcessHopSizeInFrames [private]**

Definition at line 298 of file Segmentation.h.

**6.2.6.24 zINT CSegmentation::m\_iRhythmWindowLength [private]**

Definition at line 298 of file Segmentation.h.

**6.2.6.25 zINT CSegmentation::m\_iTextureWindowHop [private]**

Definition at line 298 of file Segmentation.h.

**6.2.6.26 zINT CSegmentation::m\_iTextureWindowLength [private]**

Definition at line 298 of file Segmentation.h.

**6.2.6.27 CRingBuffer<zfLOAT32>\* CSegmentation::m\_pCOnsetCurveBuffer [private]**

Definition at line 324 of file Segmentation.h.

**6.2.6.28 CRingBuffer<zfLOAT32>\* CSegmentation::m\_pCRingBuffer [private]**

Definition at line 321 of file Segmentation.h.

**6.2.6.29 zfLOAT32 \* CSegmentation::m\_pfLogFreqs [private]**

Definition at line 313 of file Segmentation.h.

**6.2.6.30 zfLOAT32 \* CSegmentation::m\_pfMFCCBuffer [private]**

Definition at line 313 of file Segmentation.h.

**6.2.6.31 zfLOAT \* CSegmentation::m\_pfOmega [private]**

Definition at line 340 of file Segmentation.h.

**6.2.6.32 zfLOAT\* CSegmentation::m\_pfPeakSteadiness [private]**

Definition at line 340 of file Segmentation.h.

**6.2.6.33 zfLOAT32 \* CSegmentation::m\_pfPreEmphasis [private]**

Definition at line 313 of file Segmentation.h.

**6.2.6.34 zfLOAT32 \*\* CSegmentation::m\_ppfDCTCoeffs [private]**

Definition at line 313 of file Segmentation.h.

**6.2.6.35 zfLOAT32\*\* CSegmentation::m\_ppfFeatures [private]**

Definition at line 313 of file Segmentation.h.

**6.2.6.36 zfLOAT32 \*\* CSegmentation::m\_ppfMFCCFilters [private]**

Definition at line 313 of file Segmentation.h.

**6.2.6.37 SegmentationResult\_t\* CSegmentation::m\_pstResult [private]**

Definition at line 328 of file Segmentation.h.

**6.2.6.38 std::string CSegmentation::m\_strTxtFilePath [private]**

Definition at line 330 of file Segmentation.h.

The documentation for this class was generated from the following file:

- [Segmentation.h](#)

### 6.3 CTraining Class Reference

```
#include <Training.h>
```

#### Public Member Functions

- [CTraining \(int iNumOfClasses, Classifier\\_t eClassifier\)](#)
- virtual ~[CTraining \(\)](#)
- [zError\\_t AppendFeatureData \(zFLOAT \\*\\*ppfFeatureMatrix, zINT \\*piMatrixDimensions, int iClassIdx\)](#)
- [zError\\_t GetTrainingSetInfo \(TrainingSetInfo\\_t \\*pInfo\)](#)
- [Classifier\\_t GetClassifierType \(\)](#)
- [zError\\_t Process \(\)](#)
- [zError\\_t GetResultDimension \(TrainingResults\\_t eResultIdx, int \\*piResultDimensions\)](#)
- [zError\\_t GetResult \(TrainingResults\\_t eResultIdx, float \\*\\*ppfResult, const int \\*piResultDimensions\)](#)

#### Static Public Member Functions

- static [zError\\_t CreateInstance \(CTraining \\*&pCTraining, int iNumOfClasses=2, Classifier\\_t eClassifier=kClassifierLDA\)](#)
- static [zError\\_t DestroyInstance \(CTraining \\*&pCTraining\)](#)
- static const int [GetVersion \(const Version\\_t eVersionIdx\)](#)
- static const char \* [GetBuildDate \(\)](#)

#### Private Member Functions

- [zError\\_t ProcessLda \(\)](#)
- [zError\\_t ProcessQda \(\)](#)

#### Private Attributes

- [Classifier\\_t m\\_eClassifier](#)
- [CMatrix \\*\\* m\\_ppCFeatureMatrix](#)
- [CMatrix \\* m\\_apCResults \[kNumOfTrainResults\]](#)
- [zINT m\\_iNumOfClasses](#)

#### 6.3.1 Detailed Description

Definition at line 63 of file Training.h.

### 6.3.2 Constructor & Destructor Documentation

6.3.2.1 `zError_t CTraining::CTraining( int iNumOfClasses, Classifier_t eClassifier )`

6.3.2.2 `virtual CTraining::~CTraining( ) [virtual]`

### 6.3.3 Member Function Documentation

6.3.3.1 `zError_t CTraining::AppendFeatureData( zFLOAT ** ppfFeatureMatrix, zINT * piMatrixDimensions, int iClassIdx )`

6.3.3.2 `static zError_t CTraining::CreateInstance( CTraining *& pCTraining, int iNumOfClasses = 2, Classifier_t eClassifier = kClassifierLDA ) [static]`

6.3.3.3 `static zError_t CTraining::DestroyInstance( CTraining *& pCTraining ) [static]`

6.3.3.4 `static const char* CTraining::GetBuildDate( ) [static]`

6.3.3.5 `Classifier_t CTraining::GetClassifierType( ) [inline]`

Definition at line 80 of file Training.h.

References m\_eClassifier.

```
{return m_eClassifier;};
```

6.3.3.6 `zError_t CTraining::GetResult( TrainingResults_t eResultIdx, float ** ppfResult, const int * piResultDimensions )`

6.3.3.7 `zError_t CTraining::GetResultDimension( TrainingResults_t eResultIdx, int * piResultDimensions )`

6.3.3.8 `zError_t CTraining::GetTrainingSetInfo( TrainingSetInfo_t * pInfo )`

6.3.3.9 `static const int CTraining::GetVersion( const Version_t eVersionIdx ) [static]`

6.3.3.10 `zError_t CTraining::Process( )`

6.3.3.11 `zError_t CTraining::ProcessLda( ) [private]`

6.3.3.12 `zError_t CTraining::ProcessQda( ) [private]`

### 6.3.4 Field Documentation

6.3.4.1 `CMatrix* CTraining::m_apCResults[kNumOfTrainResults] [private]`

Definition at line 95 of file Training.h.

**6.3.4.2 Classifier\_t CTraining::m\_eClassifier [private]**

Definition at line 92 of file Training.h.

Referenced by GetClassifierType().

**6.3.4.3 zINT CTraining::m\_iNumberOfClasses [private]**

Definition at line 97 of file Training.h.

**6.3.4.4 CMATRIX\*\* CTraining::m\_ppCFeatureMatrix [private]**

Definition at line 93 of file Training.h.

The documentation for this class was generated from the following file:

- [Training.h](#)

## 6.4 FeatureSimilarityResult\_t\_tag Struct Reference

```
#include <FeatureSimilarity_C.h>
```

### Data Fields

- float fMaxLikelihood
- int iFeatureMatrixIndex

### 6.4.1 Detailed Description

Definition at line 49 of file FeatureSimilarity\_C.h.

### 6.4.2 Field Documentation

**6.4.2.1 float FeatureSimilarityResult\_t\_tag::fMaxLikelihood**

Definition at line 51 of file FeatureSimilarity\_C.h.

**6.4.2.2 int FeatureSimilarityResult\_t\_tag::iFeatureMatrixIndex**

Definition at line 52 of file FeatureSimilarity\_C.h.

The documentation for this struct was generated from the following file:

- [FeatureSimilarity\\_C.h](#)

## 6.5 SegmentationResult\_t\_tag Struct Reference

```
#include <Segmentation_C.h>
```

### Data Fields

- float **iStartTimeInS**  
*segment start*
- float **iStopTimeInS**  
*segment stop*
- SegmentationClasses\_t **eEstimatedClass**  
*class*
- float **fEstimationReliability**  
*estimate of result reliability*

#### 6.5.1 Detailed Description

declaration of result structure

Definition at line 24 of file Segmentation\_C.h.

#### 6.5.2 Field Documentation

##### 6.5.2.1 SegmentationClasses\_t SegmentationResult\_t\_tag::eEstimatedClass

class

Definition at line 29 of file Segmentation\_C.h.

##### 6.5.2.2 float SegmentationResult\_t\_tag::fEstimationReliability

estimate of result reliability

Definition at line 31 of file Segmentation\_C.h.

##### 6.5.2.3 float SegmentationResult\_t\_tag::iStartTimeInS

segment start

Definition at line 26 of file Segmentation\_C.h.

##### 6.5.2.4 float SegmentationResult\_t\_tag::iStopTimeInS

segment stop

Definition at line 26 of file Segmentation\_C.h.

The documentation for this struct was generated from the following file:

- Segmentation\_C.h

## 6.6 TrainingSetInfo\_t\_tag Struct Reference

info structure on the training data

```
#include <Training_C.h>
```

### Data Fields

- float **afPrior** [kMaxNumOfClasses]  
*probability of class in the training set*
- int **iNumOfFeatures**  
*number of features*
- int **iOverallNumOfObservations**  
*number of observations for all classes*

#### 6.6.1 Detailed Description

info structure on the training data

Definition at line 59 of file Training\_C.h.

#### 6.6.2 Field Documentation

##### 6.6.2.1 float TrainingSetInfo\_t\_tag::afPrior[kMaxNumOfClasses]

probability of class in the training set

Definition at line 61 of file Training\_C.h.

##### 6.6.2.2 int TrainingSetInfo\_t\_tag::iNumOfFeatures

number of features

Definition at line 62 of file Training\_C.h.

##### 6.6.2.3 int TrainingSetInfo\_t\_tag::iOverallNumOfObservations

number of observations for all classes

Definition at line 62 of file Training\_C.h.

The documentation for this struct was generated from the following file:

- [Training\\_C.h](#)

## 7 File Documentation

### 7.1 Classification\_C.h File Reference

C-wrapper for the CClassification class.

```
#include "Globals.h" Include dependency graph for Classification_C.h:
```

#### Data Structures

- struct [ClassificationResult\\_t\\_tag](#)

### Typedefs

- `typedef struct ClassificationResult_t_tag ClassificationResult_t`

### Functions

- `zError_t Class_CreateInstance` (void \*\*pphClassificationHandle, `Classifier_t` e-  
Classifier=`kClassifierLDA`, float fHopSizeInS=.5F, float fWindowSizeInS=2.0F)
- `zError_t Class_DestroyInstance` (void \*\*pphClassificationHandle)
- const int `Class_GetVersion` (const `Version_t` eVersionIdx)
- const char \* `Class_GetBuildDateString` ()
- `zError_t Class_SetParamPath2TrainResults` (void \*phClassificationHandle, char  
\*pcTxtFilePath)
- `zError_t Class_SetParamTrainResults` (void \*phClassificationHandle, float \*\*ppf-  
TrainResult, int iNumRows, int iNumCols, `TrainingResults_t` eTrainResult)
- `zError_t Class_CalcSegments` (void \*phClassificationHandle, bool bUseAPosteriori-  
Probability=false)
- `zError_t Class_RawClassification` (void \*phClassificationHandle, float \*pfUn-  
NormedResults, float \*pfObservation, int iLengthOfObservation, bool bUseA-  
PosterioriProbability=false)
- `zError_t Class_RawPostProcess` (void \*phClassificationHandle, float \*\*ppfProbabilities,  
int iNumOfProbabilities, bool bUseAPosterioriProbability=false)
- int `Class.GetSizeOfResult` (void \*phClassificationHandle)
- `zError_t Class_GetResult` (void \*phClassificationHandle, `ClassificationResult_t`  
\*pstResult)
- `zError_t Class_SetFeatureMatrix` (void \*phClassificationHandle, float \*\*ppfFeature-  
Matrix, int iRows, int iColumns, float fFirstTimeStampInS=0, bool bDontAlloc-  
InternalMemory=false)
- `zError_t Class_SetParamAPrioriProbabilities` (void \*phClassificationHandle, float  
\*pfAPrioriProbabilities, int iNumOfClasses)
- `zError_t Class_SetParamTransitionWeight` (void \*phClassificationHandle, float  
fTransitionWeight=.6F)
- `zError_t Class_SetParamMinClassTime` (void \*phClassificationHandle, float f-  
ClassTimeInS=1.0F)
- `zError_t Class_SetParamSetClassificationByFeatureLThresh` (void \*phClassification-  
Handle, int iFeatureIdx=0, float fFeatureThresh=.1F, int iClassIdx=0, float f-  
ClassProb=.8F)

#### 7.1.1 Detailed Description

C-wrapper for the CClassification class. :

Definition in file [Classification\\_C.h](#).

#### 7.1.2 Typedef Documentation

##### 7.1.2.1 `typedef struct ClassificationResult_t_tag ClassificationResult_t`

declaration of result structure

### 7.1.3 Function Documentation

**7.1.3.1 zError\_t Class\_CalcSegments ( void \* *phClassificationHandle*, bool *bUseAPosterioriProbability* = false )**

calculates result (combined call of Class\_RawClassification and Class\_CalcPostProcess)

#### Parameters

<i>ph-Classification-Handle</i>	: handle to the instance
<i>bUseAPosterioriProbability</i>	: bool indicating if for internal processing the a posteriori probability should be used (true) or the likelihood should be used (false)

#### Returns

int : 0 if no error

**7.1.3.2 zError\_t Class.CreateInstance ( void \*\* *pphClassificationHandle*, Classifier\_t *eClassifier* = kClassifierLDA, float *fHopSizeInS* = .5F, float *fWindowSizeInS* = 2.0F )**

Creates a new instance of the segmentation lib

#### Parameters

<i>pph-Classification-Handle</i>	: handle to the new instance
<i>eClassifier</i>	: classifier used (see Classifier_t)
<i>fHopSizeInS</i>	: hop size in seconds used by feature calculation
<i>fWindowSizeInS</i>	: texture window size in seconds used by feature calculation

#### Returns

int : 0 if no error

**7.1.3.3 zError\_t Class\_DestroyInstance ( void \*\* *pphClassificationHandle* )**

destroys a previously created instance of the segmentation lib

#### Parameters

<i>pph-Classification-Handle</i>	: handle to the instance
----------------------------------	--------------------------

**Returns**

int : 0 if no error

**7.1.3.4 const char\* Class\_GetBuildDateString( )**

returns a string containing the build date of this library

**Returns**

char\* : string

**7.1.3.5 zError\_t Class\_GetResult( void \* phClassificationHandle,  
ClassificationResult\_t \* pstResult )**

copies the result to pstResult

**Parameters**

<i>ph-</i> <i>Classification-</i> <i>Handle</i>	: handle to the instance
<i>*pstResult</i>	: pointer to allocated memory where the result can be copied into

**Returns**

int : 0 if no error

**7.1.3.6 int Class.GetSizeOfResult( void \* phClassificationHandle )**

returns the size of the result vector (as number of structs)

**Parameters**

<i>ph-</i> <i>Classification-</i> <i>Handle</i>	: handle to the instance
---	--------------------------

**Returns**

int : number of results

**7.1.3.7 const int Class\_GetVersion( const Version\_t eVersionIdx )**

returns a string containing the version number of this library

**Returns**

char\* : string

---

**7.1.3.8 zError\_t Class\_RawClassification ( void \* *phClassificationHandle*, float \* *pfUnNormedResults*, float \* *pfObservation*, int *iLengthOfObservation*, bool *bUseAPosterioriProbability* = false )**

calculates result for one observation (no normalization, no thresholding, etc...)

#### Parameters

<i>ph-Classification-Handle</i>	: handle to the instance
<i>pfUn-Normed-Results</i>	: result of dimension kNumClasses is written to this buffer (note that the result is "flipped" --> the lower the value, the higher the probability)
<i>pf-Observation</i>	: input feature vector of length kNumFeatures
<i>iLengthOf-Observation</i>	: has to equal kNumFeatures
<i>bUseAPosteriori-Probability</i>	: bool indicating if for internal processing the a posteriori probability should be used (true) or the likelihood should be used (false)

#### Returns

int : 0 if no error

**7.1.3.9 zError\_t Class\_RawPostProcess ( void \* *phClassificationHandle*, float \*\* *ppfProbabilities*, int *iNumOfProbabilities*, bool *bUseAPosterioriProbability* = false )**

calculates the final classification after the probabilities have been computed (e.g. by Class\_RawClassification)

#### Parameters

<i>ph-Classification-Handle</i>	: handle to the instance
<i>ppf-Probabilities</i>	: pointer to matrix that contains the results of Class_RawClassification (dimension: classes x observations)
<i>iNumOf-Probabilities</i>	: number of columns in matrix ppfProbabilities (equals number of observations)
<i>bUseAPosteriori-Probability</i>	: bool indicating if for internal processing the a posteriori probability has been used

#### Returns

int : 0 if no error

---

**7.1.3.10 zError\_t Class\_SetFeatureMatrix ( void \* *phClassificationHandle*, float \*\* *ppfFeatureMatrix*, int *iRows*, int *iColumns*, float *fFirstTimeStampInS* = 0, bool *bDontAllocInternalMemory* = false )**

sets the internal feature data

#### Parameters

<i>ph-Classification-Handle</i>	: handle to the instance
<i>*ppf-Feature-Matrix</i>	: two dimensional matrix with intermediate results
<i>iRows</i>	: number of rows of matrix
<i>iColumns</i>	: number of columns of matrix
<i>fFirstTime-StampInS</i>	: time stamp indicating the start time of the block the first feature has been calculated from
<i>bDontAlloc-Internal-Memory</i>	: flag (should the processing be done on the externally allocated memory, or should the matrix be copied)

#### Returns

int : 0 if no error

**7.1.3.11 zError\_t Class\_SetParamAPrioriProbabilities ( void \* *phClassificationHandle*, float \* *pfAPrioriProbabilities*, int *iNumberOfClasses* )**

sets the a priori probabilities of each class for the classification process

#### Parameters

<i>ph-Classification-Handle</i>	: handle to the instance
<i>*pfAPriori-Probabilities</i>	: a priori probabilities of all classes (sum has to equal 1.0F)
<i>iNumberOf-Classes</i>	: length of buffer pfAPrioriProbabilities

#### Returns

int : 0 if no error

**7.1.3.12 zError\_t Class\_SetParamMinClassTime ( void \* *phClassificationHandle*, float *fClassTimeInS* = 1.0F )**

sets the minimum time to be spend in one class (for succeeding blocks of features)

**Parameters**

<i>ph-Classification-Handle</i>	: handle to the instance
<i>*fClass-TimeInS</i>	: discard class decisions that lead to a class time in seconds that is lower than this value

**Returns**

int : 0 if no error

**7.1.3.13 zError\_t Class\_SetParamPath2TrainResults ( void \*  
phClassificationHandle, char \* pcTxtFilePath )**

set path to input txt files (means.txt and scale.txt); call alternative to Class\_SetParam-TrainResults

**Parameters**

<i>ph-Classification-Handle</i>	: handle to the instance
<i>pcTxtFilePath</i>	: path

**Returns**

int : 0 if no error

**7.1.3.14 zError\_t Class\_SetParamSetClassificationByFeatureLThresh ( void \*  
phClassificationHandle, int iFeatureIdx = 0, float fFeatureThresh = .1F, int  
iClassIdx = 0, float fClassProb = .8F )**

sets the feature and values that force a classification result for all feature values below this threshold

**Parameters**

<i>ph-Classification-Handle</i>	: handle to the instance
<i>iFeatureIdx</i>	: the index of the feature to be used
<i>fFeature-Thresh</i>	: set the class probability for all blocks where the feature is lower than this threshold
<i>iClassIdx</i>	: set this class to this special result
<i>fClassProb</i>	: probability the class with iClassIdx is set to, the other classes are set to (1-fClassProb)/NumOfClasses

**Returns**

int : 0 if no error

**7.1.3.15 zError\_t Class\_SetParamTrainResults ( void \* *phClassificationHandle*, float \*\* *ppfTrainResult*, int *iNumRows*, int *iNumCols*, TrainingResults\_t *eTrainResult* )**

set train result matrixes (means and scale); call alternative to Class\_SetParamPath2-TrainResults; it is imperative to call this function with TrainingResults\_t::kTrainResult-Means first to ensure correct internal initialization

**Parameters**

<i>ph-Classification-Handle</i>	: handle to the instance
<i>ppfTrain-Result</i>	: matrix containing the training result as specified in <i>eTrainResult</i> , matrix dimensions ([rows][cols])
<i>iNumRows</i>	: number of matrix rows
<i>iNumCols</i>	: number of matrix cols
<i>eTrainResult</i>	: which result is currently handed over (see <a href="#">TrainingResults_t</a> )

**Returns**

int : 0 if no error

**7.1.3.16 zError\_t Class\_SetParamTransitionWeight ( void \* *phClassificationHandle*, float *fTransitionWeight* = .6F )**

sets the internal cost of jumping from one class to another

**Parameters**

<i>ph-Classification-Handle</i>	: handle to the instance
<i>*f-Transition-Weight</i>	: make the transition between classes more difficult by applying this weight

**Returns**

int : 0 if no error

## 7.2 FeatureExtraction\_C.h File Reference

C-wrapper for the CFeatureExtraction class.

```
#include "Globals.h" Include dependency graph for FeatureExtraction_C.h:
```

## Functions

- `zError_t FeatEx_CreateInstance` (`void **pphFeatureExtractionHandle, int iSampleRate, int iNumberOfChannels, float fWindowLengthInS=2.0F, float fHopLengthInS=0.5F`)
- `zError_t FeatEx_DestroyInstance` (`void **pphFeatureExtractionHandle`)
- `const int FeatEx_GetVersion` (`const Version_t eVersionIdx`)
- `const char * FeatEx_GetBuildDateString` ()
- `float FeatEx_GetFeatureHopSizeInS` (`void *phFeatureExtractionHandle`)
- `float FeatEx_GetFeatureWindowSizeInS` (`void *phFeatureExtractionHandle`)
- `int FeatEx_GetNumOfFeatures` (`void *phFeatureExtractionHandle`)
- `const char * FeatEx_GetMainFeatureName` (`void *phFeatureExtractionHandle, int iFeatureIdx`)
- `const char * FeatEx_GetSubFeatureName` (`void *phFeatureExtractionHandle, int iFeatureIdx`)
- `zError_t FeatEx_Initialize` (`void *phFeatureExtractionHandle, int iOverallInputFileLengthInFrames, bool bStoreIntermediateResults`)
- `zError_t FeatEx_PreProcess` (`void *phFeatureExtractionHandle, float *pfInputBufferInterleaved, int iNumberOfFrames`)
- `zError_t FeatEx_Process` (`void *phFeatureExtractionHandle, float *pfInputBufferInterleaved, int iNumberOfFrames`)
- `zError_t FeatEx_FinishProcess` (`void *phFeatureExtractionHandle`)
- `zError_t FeatEx.GetSizeOfFeatureMatrix` (`void *phFeatureExtractionHandle, int *piRows, int *piColumns`)
- `zError_t FeatEx_GetFeatureMatrix` (`void *phFeatureExtractionHandle, float **ppfFeatureMatrix`)
- `float ** FeatEx_GetFeatureMatrixPointer` (`void *phFeatureExtractionHandle`)
- `zError_t FeatEx.GetSizeOfIntermediateResult` (`void *phFeatureExtractionHandle, int *piRows, int *piColumns`)
- `zError_t FeatEx_GetIntermediateResult` (`void *phFeatureExtractionHandle, float **ppfFeatureMatrix`)
- `float ** FeatEx_GetIntermediateResultPointer` (`void *phFeatureExtractionHandle`)
- `float FeatEx_GetFirstTimeStamp` (`void *phFeatureExtractionHandle`)

### 7.2.1 Detailed Description

C-wrapper for the CFeatureExtraction class. :

Definition in file [FeatureExtraction\\_C.h](#).

### 7.2.2 Function Documentation

#### 7.2.2.1 `zError_t FeatEx_CreateInstance ( void **pphFeatureExtractionHandle, int iSampleRate, int iNumberOfChannels, float fWindowLengthInS = 2.0F, float fHopLengthInS = 0.5F )`

Creates a new instance of the segmentation lib

**Parameters**

<i>pphFeature-Extraction-Handle</i>	: handle to the new instance
<i>iSampleRate</i>	: sample rate of audio file
<i>iNumberOfChannels</i>	: number of audio channels
<i>fWindowLengthInS</i>	: length of window in seconds for each observation (must be multiple of 0.01s, must be longer than fHopLengthInS)
<i>fHopLengthInS</i>	: length of window hop in seconds (must be multiple of 0.01s)

**Returns**

int : 0 if no error

**7.2.2.2 zError\_t FeatEx\_DestroyInstance ( void \*\* *pphFeatureExtractionHandle* )**

destroys a previously created instance of the segmentation lib

**Parameters**

<i>pphFeature-Extraction-Handle</i>	: handle to the instance
-------------------------------------	--------------------------

**Returns**

int : 0 if no error

**7.2.2.3 zError\_t FeatEx\_FinishProcess ( void \* *phFeatureExtractionHandle* )**

called to signal there is no more audio data available

**Parameters**

<i>phFeature-Extraction-Handle</i>	: handle to the instance
------------------------------------	--------------------------

**Returns**

int : 0 if no error

**7.2.2.4 const char\* FeatEx\_GetBuildDateString ( )**

returns a string containing the build date of this library

**Returns**

char\* : string

**7.2.2.5 float FeatEx\_GetFeatureHopSizeInS ( void \* *phFeatureExtractionHandle* )**

returns the feature hopsize in seconds

**Parameters**

<i>phFeature-Extraction-Handle</i>	: handle to the instance
------------------------------------	--------------------------

**Returns**

float : feature hopsize in seconds

**7.2.2.6 zError\_t FeatEx\_GetFeatureMatrix ( void \* *phFeatureExtractionHandle*, float \*\* *ppfFeatureMatrix* )**

copies the internal intermediate feature data

**Parameters**

<i>phFeature-Extraction-Handle</i>	: handle to the instance
<i>**ppf-Feature-Matrix</i>	: two-dimensional matrix with the size from FeatEx.GetSizeOfFeatureMatrix

**Returns**

int : 0 if no error

**7.2.2.7 float\*\* FeatEx\_GetFeatureMatrixPointer ( void \* *phFeatureExtractionHandle* )**

returns the pointer to the internal intermediate feature data

**Parameters**

<i>phFeature-Extraction-Handle</i>	: handle to the instance
------------------------------------	--------------------------

**Returns**

float\*\* : pointer to result (do \*not\* free!)

**7.2.2.8 float FeatEx\_GetFeatureWindowSizeInS ( void \*  
phFeatureExtractionHandle )**

returns the feature windowsize in seconds

**Parameters**

<i>phFeature- Extraction- Handle</i>	: handle to the instance
--	--------------------------

**Returns**

float : feature windowsize in seconds

**7.2.2.9 float FeatEx\_GetFirstTimeStamp ( void \* phFeatureExtractionHandle )**

returns the timestamp for the first observation in seconds

**Parameters**

<i>phFeature- Extraction- Handle</i>	: handle to the instance
--	--------------------------

**Returns**

float : first timestamp

**7.2.2.10 zError\_t FeatEx\_GetIntermediateResult ( void \*  
phFeatureExtractionHandle, float \*\* ppfFeatureMatrix )**

returns the internal feature data

**Parameters**

<i>phFeature- Extraction- Handle</i>	: handle to the instance
<i>**ppf- Feature- Matrix</i>	: two-dimensional matrix with the size from FeatEx_GetSizeOf- FeatureMatrix

**Returns**

int : 0 if no error

**7.2.2.11 float\*\* FeatEx\_GetIntermediateResultPointer ( void \*  
phFeatureExtractionHandle )**

returns the pointer to the internal intermediate feature data

**Parameters**

<i>phFeature- Extraction- Handle</i>	: handle to the instance
--	--------------------------

**Returns**

float\*\* : pointer to result (do \*not\* free!)

**7.2.2.12 const char\* FeatEx\_GetMainFeatureName ( void \*  
phFeatureExtractionHandle, int iFeatureIdx )**

returns the name of a main feature

**Parameters**

<i>phFeature- Extraction- Handle</i>	: handle to the instance
<i>iFeatureIdx</i>	: feature index

**Returns**

const char : main feature name

**7.2.2.13 int FeatEx\_GetNumOfFeatures ( void \* phFeatureExtractionHandle )**

returns the overall number of features per observation

**Parameters**

<i>phFeature- Extraction- Handle</i>	: handle to the instance
--	--------------------------

**Returns**

int : main feature name

---

**7.2.2.14 zError\_t FeatEx\_GetSizeOfFeatureMatrix ( void \*  
phFeatureExtractionHandle, int \* piRows, int \* piColumns )**

returns the size of result after FinishProcess (if intermediate results should be stored for later usage)

#### Parameters

<i>phFeature-Extraction-Handle</i>	: handle to the instance
<i>*piRows</i>	: number of rows of result matrix (to be written)
<i>*piColumns</i>	: number of columns of result matrix (to be written)

#### Returns

int : 0 if no error

**7.2.2.15 zError\_t FeatEx\_GetSizeOfIntermediateResult ( void \*  
phFeatureExtractionHandle, int \* piRows, int \* piColumns )**

returns the size of intermediate result after FinishProcess (if intermediate results should be stored for later usage)

#### Parameters

<i>phFeature-Extraction-Handle</i>	: handle to the instance
<i>*piRows</i>	: number of rows of result matrix (to be written)
<i>*piColumns</i>	: number of columns of result matrix (to be written)

#### Returns

int : 0 if no error

**7.2.2.16 const char\* FeatEx\_GetSubFeatureName ( void \*  
phFeatureExtractionHandle, int iFeatureIdx )**

returns the name of a sub feature

#### Parameters

<i>phFeature-Extraction-Handle</i>	: handle to the instance
<i>iFeatureIdx</i>	: feature index

**Returns**

const char : sub feature name

**7.2.2.17 const int FeatEx\_GetVersion ( const Version\_t eVersionIdx )**

returns a string containing the version number of this library

**Returns**

char\* : string

**7.2.2.18 zError\_t FeatEx\_Initialize ( void \* phFeatureExHandle, int iOverallInputFileLengthInFrames, bool bStoreIntermediateResults )**

initializes an instance of the segmentation lib

**Parameters**

<i>phFeature-Extraction-Handle</i>	: handle to the instance
<i>iOverall-InputFile-LengthIn-Frames</i>	: number of audio frames
<i>bStore-Intermediate-Results</i>	: bool to indicate access is required to the intermediate results (keep at false if you don't know what this means)

**Returns**

int : 0 if no error

**7.2.2.19 zError\_t FeatEx\_PreProcess ( void \* phFeatureExHandle, float \* pfInputBufferInterleaved, int iNumberOfFrames )**

preprocessing loop of the audio data

**Parameters**

<i>phFeature-Extraction-Handle</i>	: handle to the instance
<i>*pfInput-Buffer-Interleaved</i>	: audio input data in pcm interleaved format
<i>iNumberOf-Frames</i>	: number of audio frames in buffer

**Returns**

int : 0 if no error

**7.2.2.20 zError\_t FeatEx\_Process ( void \* *phFeatureExtractionHandle*, float \* *pfInputBufferInterleaved*, int *iNumberOfFrames* )**

processes the audio data

**Parameters**

<i>phFeature-Extraction-Handle</i>	: handle to the instance
<i>*pfInput-Buffer-Interleaved</i>	: audio input data in pcm interleaved format
<i>iNumberOfFrames</i>	: number of audio frames in buffer

**Returns**

int : 0 if no error

## 7.3 featuresimilarity.txt File Reference

### 7.4 FeatureSimilarity\_C.h File Reference

C-wrapper for the CFeatureSimilarity class.

```
#include "Globals.h" Include dependency graph for FeatureSimilarity_C.h:
```

#### Data Structures

- struct [FeatureSimilarityResult\\_t\\_tag](#)

#### TypeDefs

- typedef struct [FeatureSimilarityResult\\_t\\_tag](#) [FeatureSimilarityResult\\_t](#)

#### Functions

- [zError\\_t FeatSim\\_CreateInstance](#) (void \*\*pphFeatureSimilarityHandle)
- [zError\\_t FeatSim\\_DestroyInstance](#) (void \*\*pphFeatureSimilarityHandle)
- const int [FeatSim\\_GetVersion](#) (const [Version\\_t](#) eVersionIdx)
- const char \* [FeatSim\\_GetBuildDateString](#) ()
- [zError\\_t FeatSim\\_Initialize](#) (void \*phFeatureSimilarityHandle, float \*pfMeans, int aiDim1[2], int aiDim2[2])

- `zError_t FeatSim_Process` (void \*phFeatureSimilarityHandle, float \*\*ppfFeatureMatrix1, float \*\*ppfFeatureMatrix2)
- `zError_t FeatSim_GetResult` (void \*phFeatureSimilarityHandle, [FeatureSimilarityResult\\_t](#) \*pstResult)

#### 7.4.1 Detailed Description

C-wrapper for the CFeatureSimilarity class. :

Definition in file [FeatureSimilarity\\_C.h](#).

#### 7.4.2 Typedef Documentation

##### 7.4.2.1 `typedef struct FeatureSimilarityResult_t_tag FeatureSimilarityResult_t`

#### 7.4.3 Function Documentation

##### 7.4.3.1 `zError_t FeatSim.CreateInstance ( void ** pphFeatureSimilarityHandle )`

Creates a new instance of the segmentation lib

#### Parameters

<code>pphFeature-Similarity-Handle</code>	: handle to the new instance
---	------------------------------

#### Returns

int : 0 if no error

##### 7.4.3.2 `zError_t FeatSim_DestroyInstance ( void ** pphFeatureSimilarityHandle )`

destroys a previously created instance of the segmentation lib

#### Parameters

<code>pphFeature-Similarity-Handle</code>	: handle to the instance
---	--------------------------

#### Returns

int : 0 if no error

##### 7.4.3.3 `const char* FeatSim_GetBuildDateString ( )`

returns a string containing the build date of this library

**Returns**

char\* : string

**7.4.3.4 zError\_t FeatSim\_GetResult ( void \* phFeatureSimilarityHandle,  
FeatureSimilarityResult\_t \* pstResult )**

processes the audio data

**Parameters**

<i>phFeature-Similarity-Handle</i>	: handle to the instance
<i>*pstResult</i>	: structure with the results (to be written)

**Returns**

int : 0 if no error

**7.4.3.5 const int FeatSim\_GetVersion ( const Version\_t eVersionIdx )**

returns a string containing the version number of this library

**Returns**

char\* : string

**7.4.3.6 zError\_t FeatSim\_Initialize ( void \* phFeatureSimilarityHandle, float \* pfMeans, int aiDim1[2], int aiDim2[2] )**

initializes an instance of the segmentation lib

**Parameters**

<i>phFeature-Similarity-Handle</i>	: handle to the instance
<i>pfMeans</i>	: vector containing the means of each individual feature (from the training set, length is aiDim1[0])
<i>aiDim1</i>	: dimensions of the first matrix, rows:features, columns: observations
<i>aiDim2</i>	: dimensions of the first matrix, rows:features, columns: observations (note that aiDim1[0] == aiDim2[0] and aiDim1[1] > aiDim2[1])

**Returns**

int : 0 if no error

**7.4.3.7 zError\_t FeatSim\_Process ( void \* phFeatureSimilarityHandle, float \*\* ppfFeatureMatrix1, float \*\* ppfFeatureMatrix2 )**

processes the audio data

**Parameters**

<i>phFeature-Similarity-Handle</i>	: handle to the instance
<i>*ppfFeature-Matrix1</i>	: big feature matrix [features x observations]
<i>ppfFeature-Matrix2</i>	: small feature matrix [features x observations]

**Returns**

int : 0 if no error

**7.5 Globals.h File Reference**

some global constants/types.

This graph shows which files directly or indirectly include this file:

**Defines**

- #define [kMaxNumberOfClasses](#) 10

**TypeDefs**

- typedef enum [Version\\_t\\_tag](#) [Version\\_t](#)  
*indices of available classifiers*
- typedef enum [Classifier\\_t\\_tag](#) [Classifier\\_t](#)  
*indices of training results*
- typedef enum [TrainingResults\\_t\\_tag](#) [TrainingResults\\_t](#)  
*definition of error types*

**Enumerations**

- enum [Version\\_t\\_tag](#) { [kMajor](#), [kMinor](#), [kPatch](#), [kRevision](#), [kNumVersionInts](#) }

- enum `Classifier_t_tag` { `kClassifierLDA`, `kClassifierQDA`, `kNumOfClassifiers` }  
*indices of available classifiers*
- enum `TrainingResults_t_tag` { `kTrainResultMeans`, `kTrainResultScale`, `kTrainResultDet`, `kNumOfTrainResults` }  
*indices of training results*
- enum `zError_t_tag` { `kMInoError`, `kMInothingToDo`, `kMInotInitialized`, `kMIMatrixDimensionMismatch`, `kMInvalidPointer`, `kMInvalidArgument`, `kMInvalidTrainingData`, `kMInufficientTrainingData`, `kMInvalidSampleRate`, `kMInvalidNumOfChannels`, `kMIFileOpenFailed`, `kMIMemAllocFailed`, `kMInternalError` }  
*definition of error types*

### 7.5.1 Detailed Description

some global constants/types. :

Definition in file [Globals.h](#).

### 7.5.2 Define Documentation

#### 7.5.2.1 `#define kMaxNumberOfClasses 10`

Definition at line 51 of file [Globals.h](#).

### 7.5.3 Typedef Documentation

#### 7.5.3.1 `typedef enum Classifier_t_tag Classifier_t`

indices of available classifiers

#### 7.5.3.2 `typedef enum TrainingResults_t_tag TrainingResults_t`

indices of training results

#### 7.5.3.3 `typedef enum Version_t_tag Version_t`

#### 7.5.3.4 `typedef enum zError_t_tag zError_t`

definition of error types

### 7.5.4 Enumeration Type Documentation

#### 7.5.4.1 `enum Classifier_t_tag`

indices of available classifiers

**Enumerator:**

*kClassifierLDA* linear discriminant analysis (default)  
*kClassifierQDA* quadratic discriminant analysis  
*kNumOfClassifiers*

Definition at line 68 of file Globals.h.

```
{  
    kClassifierLDA,  
    kClassifierQDA,  
  
    kNumOfClassifiers  
} Classifier_t;
```

**7.5.4.2 enum TrainingResults\_t\_tag**

indices of training results

**Enumerator:**

*kTrainResultMeans* index of the means matrix  
*kTrainResultScale* index of the scale matrix  
*kTrainResultDet* index of log determinant vector for each covariance matrix (only for kClassifierQDA)  
*kNumOfTrainResults*

Definition at line 80 of file Globals.h.

```
{  
    kTrainResultMeans,  
    kTrainResultScale,  
    kTrainResultDet,  
  
    kNumOfTrainResults  
} TrainingResults_t;
```

**7.5.4.3 enum Version\_t\_tag****Enumerator:**

*kMajor*  
*kMinor*  
*kPatch*  
*kRevision*  
*kNumVersionInts*

Definition at line 53 of file Globals.h.

```
{
    kMajor,
    kMinor,
    kPatch,
    kRevision,

    kNumVersionInts
} Version_t;
```

#### 7.5.4.4 enum zError\_t\_tag

definition of error types

**Enumerator:**

**kMlNoError** no error occurred

**kMlNothingToDo** either the parameter has already been set before, or the function call at this time was unnecessary

**kMlNotInitialized** the instance has not been initialized properly

**kMlMatrixDimensionMismatch** the matrix does not have the expected dimensions

**kMlInvalidPointer** one pointer in the function parameters points to an invalid address

**kMlInvalidArgument** one function parameter has a value that is not allowed (out of range)

**kMlInvalidTrainingData** something's wrong with the training data

**kMlInsufficientTrainingData** not enough training data

**kMlInvalidSampleRate** the sample rate does not meet the internal requirements

**kMlInvalidNumOfChannels** the number of audio channels does not meet the internal requirements

**kMlFileOpenFailed** a file could not be opened

**kMlMemAllocFailed** internal memory allocation failed

**kMlInternalError** a different error - please contact support

Definition at line 93 of file Globals.h.

```
{
    kMlNoError,

    kMlNothingToDo,
    kMlNotInitialized,

    kMlMatrixDimensionMismatch,

    kMlInvalidPointer,
    kMlInvalidArgument,

    kMlInvalidTrainingData,
    kMlInsufficientTrainingData,
```

```
    kMlInvalidSampleRate,
    kMlInvalidNumOfChannels,

    kMlFileOpenFailed,
    kMlMemAllocFailed,
    kMlInternalError

} zError_t;
```

## 7.6 Segmentation.h File Reference

interface of the [CSegmentation](#) class.

```
#include "Segmentation_C.h" #include "zplVecLib.h" #include
<string> Include dependency graph for Segmentation.h:
```

### Data Structures

- class [CSegmentation](#)

### Defines

- `#define PREFILTER`
- `#define RHYTHM`
- `#define kNumMelCoeffs (4)`
- `#define kNumNotches (3)`
- `#define kNumHPFilters (1)`
- `#define kNumPreFilters (kNumHPFilters+kNumNotches)`
- `#define kCompleteNumOfFeatures (kNumFeatures * kNumSubFeatures)`

### 7.6.1 Detailed Description

interface of the [CSegmentation](#) class. :

Definition in file [Segmentation.h](#).

### 7.6.2 Define Documentation

#### 7.6.2.1 `#define kCompleteNumOfFeatures (kNumFeatures * kNumSubFeatures)`

Definition at line 292 of file Segmentation.h.

#### 7.6.2.2 `#define kNumHPFilters (1)`

Definition at line 193 of file Segmentation.h.

**7.6.2.3 #define kNumMelCoeffs (4)**

Definition at line 191 of file Segmentation.h.

**7.6.2.4 #define kNumNotches (3)**

Definition at line 192 of file Segmentation.h.

**7.6.2.5 #define kNumPreFilters (kNumHPFilters+kNumNotches)**

Definition at line 194 of file Segmentation.h.

**7.6.2.6 #define PREFILTER**

Definition at line 152 of file Segmentation.h.

**7.6.2.7 #define RHYTHM**

Definition at line 154 of file Segmentation.h.

## 7.7 Segmentation\_C.h File Reference

#include "Globals.h" Include dependency graph for Segmentation\_C.h: This graph shows which files directly or indirectly include this file:

### Data Structures

- struct [SegmentationResult\\_t\\_tag](#)

### TypeDefs

- typedef enum [SegmentationClasses\\_t\\_tag](#) [SegmentationClasses\\_t](#)
- typedef struct [SegmentationResult\\_t\\_tag](#) [SegmentationResult\\_t](#)

### Enumerations

- enum [SegmentationClasses\\_t\\_tag](#) { [kDoesNotContainMusic](#) = 0, [kContainsMusic](#) = 1, [kNumOfSegmentationClasses](#) }

### Functions

- int [Segment\\_CreateInstance](#) (void \*\*pphSegmentationHandle, int iSampleRate, int iNumberOfChannels)
- int [Segment\\_DestroyInstance](#) (void \*\*pphSegmentationHandle)
- const int [Segment\\_GetVersion](#) (const [Version\\_t](#) eVersionIdx)
- const char \* [Segment\\_GetBuildDateString](#) ()
- int [Segment\\_Initialize](#) (void \*phSegmentationHandle, int iOverallInputFileLengthInFrames)

- int **Segment\_PreProcess** (void \*phSegmentationHandle, float \*pfInputBufferInterleaved, int iNumberOfFrames)
- int **Segment\_Process** (void \*phSegmentationHandle, float \*pfInputBufferInterleaved, int iNumberOfFrames)
- int **Segment\_FinishProcess** (void \*phSegmentationHandle)
- int **Segment\_SetTxtFilePath** (void \*phSegmentationHandle, char \*pcTxtFilePath)
- int **Segment\_PostProcess** (void \*phSegmentationHandle, float fTransitionWeight=0.-6F, float fMinimumClassTimeInS=1.0F, float fAPrioriProbabilityOfMusic=0.-95F, float fMinimumEnergy=0.1F, float \*\*ppfRawResult=0)
- int **Segment.GetSizeOfResult** (void \*phSegmentationHandle)
- int **Segment.GetResult** (void \*phSegmentationHandle, **SegmentationResult\_t** \*pstResult)
- int **Segment.GetSizeOfIntermediateResult** (void \*phSegmentationHandle, int \*piRows, int \*piColumns)
- int **Segment.GetIntermediateResult** (void \*phSegmentationHandle, float \*\*ppfIntermediateResult)
- int **Segment\_SetIntermediateResult** (void \*phSegmentationHandle, float \*\*ppfIntermediateResult, int iRows, int iColumns)

### 7.7.1 Typedef Documentation

#### 7.7.1.1 **typedef enum SegmentationClasses\_t\_tag SegmentationClasses\_t**

list of the valid classes

#### 7.7.1.2 **typedef struct SegmentationResult\_t\_tag SegmentationResult\_t**

declaration of result structure

### 7.7.2 Enumeration Type Documentation

#### 7.7.2.1 **enum SegmentationClasses\_t\_tag**

list of the valid classes

**Enumerator:**

**kDoesNotContainMusic** there is no music

**kContainsMusic** there was music detected

**kNumOfSegmentationClasses**

Definition at line 14 of file Segmentation\_C.h.

```
{
    kDoesNotContainMusic      = 0,
    kContainsMusic           = 1,

    kNumOfSegmentationClasses
} SegmentationClasses_t;
```

### 7.7.3 Function Documentation

**7.7.3.1 int Segment\_CreateInstance ( void \*\* *pphSegmentationHandle*, int *iSampleRate*, int *iNumberOfChannels* )**

Creates a new instance of the segmentation lib

#### Parameters

<i>pph-Segmentation-Handle</i>	: handle to the new instance
<i>iSampleRate</i>	: sample rate of audio file
<i>iNumberOfChannels</i>	: number of audio channels

#### Returns

int : 0 if no error

**7.7.3.2 int Segment\_DestroyInstance ( void \*\* *pphSegmentationHandle* )**

destroys a previously created instance of the segmentation lib

#### Parameters

<i>pph-Segmentation-Handle</i>	: handle to the instance
--------------------------------	--------------------------

#### Returns

int : 0 if no error

**7.7.3.3 int Segment\_FinishProcess ( void \* *phSegmentationHandle* )**

called to signal there is no more audio data available

#### Parameters

<i>ph-Segmentation-Handle</i>	: handle to the instance
-------------------------------	--------------------------

#### Returns

int : 0 if no error

**7.7.3.4 const char\* Segment\_GetBuildDateString( )**

returns a string containing the build date of this library

**Returns**

char\* : string

**7.7.3.5 int Segment\_GetIntermediateResult ( void \* *phSegmentationHandle*, float \*\* *ppfIntermediateResult* )**

returns the internal feature data

**Parameters**

<i>ph-Segmentation-Handle</i>	: handle to the instance
<i>**ppf-Intermediate-Result</i>	: two-dimensional matrix with the size from GetSizeOfIntermediate-Result

**Returns**

int : 0 if no error

**7.7.3.6 int Segment\_GetResult ( void \* *phSegmentationHandle*, SegmentationResult\_t \* *pstResult* )**

copies the result to *pstResult*

**Parameters**

<i>ph-Segmentation-Handle</i>	: handle to the instance
<i>*pstResult</i>	: pointer to allocated memory where the result can be copied into

**Returns**

int : 0 if no error

**7.7.3.7 int Segment\_GetSizeOfIntermediateResult ( void \* *phSegmentationHandle*, int \* *piRows*, int \* *piColumns* )**

returns the size of intermediate result after FinishProcess (if intermediate results should be stored for later usage)

**Parameters**

<i>ph-Segmentation-Handle</i>	: handle to the instance
<i>*piRows</i>	: number of rows of result matrix (to be written)
<i>*piColumns</i>	: number of columns of result matrix (to be written)

**Returns**

int : 0 if no error

**7.7.3.8 int Segment\_GetSizeOfResult ( void \* *phSegmentationHandle* )**

returns the size of the result vector (as number of structs)

**Parameters**

<i>ph-Segmentation-Handle</i>	: handle to the instance
-------------------------------	--------------------------

**Returns**

int : 0 if no error

**7.7.3.9 const int Segment\_GetVersion ( const Version\_t *eVersionIdx* )**

returns a string containing the version number of this library

**Returns**

char\* : string

**7.7.3.10 int Segment\_Initialize ( void \* *phSegmentationHandle*, int *iOverallInputFileLengthInFrames* )**

initializes an instance of the segmentation lib

**Parameters**

<i>ph-Segmentation-Handle</i>	: handle to the instance
<i>iOverall-InputFile-LengthIn-Frames</i>	: number of audio frames

**Returns**

int : 0 if no error

**7.7.3.11 int Segment\_PostProcess ( void \* *phSegmentationHandle*, float *fTransitionWeight* = 0.6F, float *fMinimumClassTimeInS* = 1.0F, float *fAPrioriProbabilityOfMusic* = 0.95F, float *fMinimumEnergy* = 0.1F, float \*\* *ppfRawResult* = 0 )**

postprocesses the extracted data and calculates result

**Parameters**

<i>ph-Segmentation-Handle</i>	: handle to the instance
<i>fTransition-Weight</i>	: additional cost for the transition between classes (the higher the less jumps between classes), must be positive
<i>fMinimum-ClassTime-InS</i>	: minimum active time of a class
<i>fAPriori-Probability-OfMusic</i>	: if class 1 and 2 are not equally probably, change this (between 0...1)
<i>fMinimum-Energy</i>	: if a processing frame does contain an sqrt(rms)-energy below this threshold, this frame is not regarded as able to contain music (between 0...1)
<i>ppfRaw-Result</i>	: get raw classification result for each feature vector (memory has to be allocated, dimensions [kNumOfSegmentationClasses][piColumns (from Segment_GetSizeOfIntermediateResult)])

**Returns**

int : 0 if no error

**7.7.3.12 int Segment\_PreProcess ( void \* *phSegmentationHandle*, float \* *pfInputBufferInterleaved*, int *iNumberOfFrames* )**

preprocessing loop of the audio data

**Parameters**

<i>ph-Segmentation-Handle</i>	: handle to the instance
<i>*pfInput-Buffer-Interleaved</i>	: audio input data in pcm interleaved format
<i>iNumberOf-Frames</i>	: number of audio frames in buffer

**Returns**

int : 0 if no error

**7.7.3.13 int Segment\_Process ( void \* *phSegmentationHandle*, float \* *pflInputBufferInterleaved*, int *iNumberOfFrames* )**

processes the audio data

**Parameters**

<i>ph-Segmentation-Handle</i>	: handle to the instance
<i>*pflInput-Buffer-Interleaved</i>	: audio input data in pcm interleaved format
<i>iNumberOf-Frames</i>	: number of audio frames in buffer

**Returns**

int : 0 if no error

**7.7.3.14 int Segment\_SetIntermediateResult ( void \* *phSegmentationHandle*, float \*\* *ppfIntermediateResult*, int *iRows*, int *iColumns* )**

sets the internal feature data (no Segment\_Process is required then)

**Parameters**

<i>ph-Segmentation-Handle</i>	: handle to the instance
<i>**ppf-Intermediate-Result</i>	: two dimensional matrix with intermediate results
<i>iRows</i>	: number of rows of matrix
<i>iColumns</i>	: number of columns of matrix

**Returns**

int : 0 if no error

**7.7.3.15 int Segment\_SetTxtFilePath ( void \* *phSegmentationHandle*, char \* *pcTxtFilePath* )**

set path to input txt files (means.txt and scale.txt)

**Parameters**

<i>ph-Segmentation-Handle</i>	: handle to the instance
<i>pcTxtFilePath</i>	: path

**Returns**

int : 0 if no error

**7.8 Training.h File Reference**

interface of the [CTraining](#) class.

```
#include "zplVecLib.h" Include dependency graph for Training.h:
```

**Data Structures**

- class [CTraining](#)

**7.8.1 Detailed Description**

interface of the [CTraining](#) class. :

Definition in file [Training.h](#).

**7.9 Training\_C.h File Reference**

C-interface wrapper for the Training.

```
#include "Globals.h" Include dependency graph for Training_C.h:
```

**Data Structures**

- struct [TrainingSetInfo\\_t\\_tag](#)  
*info structure on the training data*

**Typedefs**

- typedef struct [TrainingSetInfo\\_t\\_tag](#) [TrainingSetInfo\\_t](#)  
*info structure on the training data*

## Functions

- `zError_t Train.CreateInstance` (void \*\*phTrainingHandle, int iNumberOfClasses, `Classifier_t` eClassifier)  
*creates a new instance of training*
- `zError_t Train_DestroyInstance` (void \*\*phTrainingHandle)  
*destroys an instance of training*
- `const int Train_GetVersion` (const `Version_t` eVersionIdx)
- `const char * Train_GetBuildDateString` ()
- `zError_t Train_AppendFeatureData` (void \*phTrainingHandle, float \*\*ppfFeatureMatrix, int \*piMatrixDimensions, int iClassIdx)  
*append new feature data to the training set*
- `zError_t Train_GetTrainingSetInfo` (void \*phTrainingHandle, `TrainingSetInfo_t` \*pInfo)  
*returns some information about the training data*
- `Classifier_t Train_GetClassifierType` (void \*phTrainingHandle)  
*returns the selected classifier of this instance*
- `zError_t Train_Process` (void \*phTrainingHandle)  
*do the training, the internal data is deleted after process*
- `zError_t Train_GetResultDimension` (void \*phTrainingHandle, `TrainingResults_t` eResultIdx, int \*piResultDimensions)  
*returns the dimension, i.e. number of columns and number of rows of the result with the given index*
- `zError_t Train_GetResult` (void \*phTrainingHandle, `TrainingResults_t` eResultIdx, float \*\*ppfResult, const int \*piResultDimensions)  
*returns the result with the specific index*

### 7.9.1 Detailed Description

C-interface wrapper for the Training. :

Definition in file [Training\\_C.h](#).

### 7.9.2 Typedef Documentation

#### 7.9.2.1 `typedef struct TrainingSetInfo_t_tag TrainingSetInfo_t`

info structure on the training data

### 7.9.3 Function Documentation

#### 7.9.3.1 `zError_t Train_AppendFeatureData ( void * phTrainingHandle, float ** ppfFeatureMatrix, int * piMatrixDimensions, int iClassIdx )`

append new feature data to the training set

**Parameters**

<i>phTrainingHandle</i>	handle to training instance
<i>ppfFeatureMatrix</i>	new data (dimensions ([iNumOfFeatures]x[iNumOfObservations])
<i>piMatrixDimensions</i>	dimensions of ppfFeatureMatrix [iRows][iColumns]
<i>iClassIdx</i>	class index (e.g. speech = 0, music = 1)

**Returns**

0 if no error

**7.9.3.2 zError\_t Train.CreateInstance ( void \*\* *pphTrainingHandle*, int *iNumberOfClasses*, Classifier\_t *eClassifier* )**

creates a new instance of training

**Parameters**

<i>pphTrainingHandle</i>	handle to new instance
<i>iNumberOfClasses</i>	number of classes to be trained for separation (2...10)
<i>eClassifier</i>	classifier to be trained (see <a href="#">Classifier_t</a> )

**Returns**

0 if no error

**7.9.3.3 zError\_t Train\_DestroyInstance ( void \*\* *pphTrainingHandle* )**

destroys an instance of training

**Parameters**

<i>pphTrainingHandle</i>	handle to instance to be destroyed
--------------------------	------------------------------------

**Returns**

0 if no error

**7.9.3.4 const char\* Train\_GetBuildDateString ( )**

returns a string containing the build date of this library

**Returns**

char\* : string

**7.9.3.5 Classifier\_t Train\_GetClassifierType ( void \* *phTrainingHandle* )**

returns the selected classifier of this instance

**Parameters**

<i>phTraining-Handle</i>	handle to training instance
--------------------------	-----------------------------

**Returns**

see [Classifier\\_t](#)

**7.9.3.6 zError\_t Train\_GetResult ( void \* *phTrainingHandle*, TrainingResults\_t *eResultIdx*, float \*\* *ppfResult*, const int \* *piResultDimensions* )**

returns the result with the specific index

**Parameters**

<i>phTraining-Handle</i>	handle to training instance
<i>eResultIdx</i>	index of result we are interested in
<i>ppfResult</i>	matrix where the result values are copied to
<i>piResult-Dimensions</i>	dimensions of ppfResult, has to equal the parameter from <a href="#">Train_GetResultDimension</a>

**Returns**

0 if no error

**See also**

[Train\\_GetResultDimension](#)

**7.9.3.7 zError\_t Train\_GetResultDimension ( void \* *phTrainingHandle*, TrainingResults\_t *eResultIdx*, int \* *piResultDimensions* )**

returns the dimension, i.e. number of columns and number of rows of the result with the given index

**Parameters**

<i>phTraining-Handle</i>	handle to training instance
--------------------------	-----------------------------

<i>eResultIdx</i>	index of result we are interested in
<i>piResult-Dimensions</i>	the result dimensions are written here ([iNumberOfRows]x[iNumberOfColumns])

**Returns**

0 if no error

**7.9.3.8 zError\_t Train\_GetTrainingSetInfo ( void \* *phTrainingHandle*, TrainingSetInfo\_t \* *pInfo* )**

returns some information about the training data

**Parameters**

<i>phTraining-Handle</i>	handle to training instance
<i>pInfo</i>	pointer to structure where the information is copied into

**Returns**

0 if no error

**7.9.3.9 const int Train\_GetVersion ( const Version\_t *eVersionIdx* )**

returns a string containing the version number of this library

**Returns**

char\* : string

**7.9.3.10 zError\_t Train\_Process ( void \* *phTrainingHandle* )**

do the training, the internal data is deleted after process

**Parameters**

<i>phTraining-Handle</i>	handle to training instance
--------------------------	-----------------------------

**Returns**

0 if no error