AudioTranskriptor

Generated by Doxygen 1.9.8

Chapter 1

Todo List

Member FileManager::findExistingMeetings () const

Diese Methode ist nur ein temporärer Platzhalter für die Demonstration und wird entfernt, sobald die Anwendung mit der Datenbank verknüpft wird.

Member FileManager::getMeetingJsonPath (const QString &meetingId, bool isEdited) const

Diese Methode ist nur ein temporärer Platzhalter für die Demonstration und wird entfernt, sobald die Anwendung mit der Datenbank verknüpft wird.

Member FileManager::getMeetingsDirectory () const

Diese Methode ist nur ein temporärer Platzhalter für die Demonstration und wird entfernt, sobald die Anwendung mit der Datenbank verknüpft wird.

Member FileManager::loadJson (const QString &filePath, bool &ok) const

Diese Methode ist nur ein temporärer Platzhalter für die Demonstration bis die Anwendung mit der Datenbank verknüpft wird. Danach kann sie aber evtl. erhalten bleiben.

Member FileManager::saveJson (const QString &filePath, const QJsonDocument &doc) const

Diese Methode ist nur ein temporärer Platzhalter für die Demonstration bis die Anwendung mit der Datenbank verknüpft wird. Danach kann sie aber evtl. erhalten bleiben.

Member MainWindow::closeEvent (QCloseEvent *event) override

Die Logik zum Speichern bei ungesicherten Änderungen soll zukünftig ebenfalls die **Datenbank-↩ Speicherfunktion** aufrufen.

Member MainWindow::loadMeetings ()

Aktuell wird das Dateisystem durchsucht. Zukünftig soll diese Methode eine **Datenbankabfrage** ausführen, um alle gespeicherten Meetings abzurufen.

Member MainWindow::loadTranscriptionFromJson ()

Diese Funktion zum Laden einer beliebigen Datei bleibt bestehen, aber die primäre Ladefunktion wird der Datenbankzugriff über onMeetingSelected().

Member MainWindow::onMeetingSelected (QListWidgetItem *item)

Die Methode verwendet derzeit den FileManager, um eine JSON-Datei zu laden. Dies soll durch eine **Datenbankabfrage** ersetzt werden, die das Meeting anhand seiner ID lädt.

Member MainWindow::onPollTranscripts ()

Diese Methode ist nur für Demonstrationszwecke und wird entfernt, sobald die Echtzeit-Verarbeitung via ASR-Manager implementiert ist.

Member MainWindow::restoreOriginalTranscription ()

Lädt aktuell das `_original.json`-File. Zukünftig soll dies den Originalzustand des Transkripts aus der **Datenbank** wiederherstellen.

Member MainWindow::saveTranscriptionToJson ()

Ersetzt aktuell die zugehörige JSON-Datei. Dies soll in Zukunft einen 'UPDATE'-Befehl an die **Datenbank** senden, um den bestehenden Eintrag zu aktualisieren.

Member MainWindow::saveTranscriptionToJsonAs ()

Statt als neue Datei zu speichern, soll dies zukünftig einen 'INSERT'-Befehl an die **Datenbank** senden, um einen neuen Eintrag zu erzeugen.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 demo Namespace Reference

Variables

- str text

6.1.1 Variable Documentation

6.1.1.1 text

```
str demo.text
```

6.2 generate_tags Namespace Reference

Functions

- generate_tags (text)

Variables

- input_text = sys.stdin.read()
- generated_tags = generate_tags(input_text)

6.2.1 Function Documentation

6.2.1.1 generate_tags()

```
generate_tags.generate_tags (
              text )

Analysiert einen deutschen Text und extrahiert zwei Arten von Tags.

Die Funktion nutzt das spaCy-Modell 'de_core_news_lg' zur Analyse. Es werden
zwei Strategien zur Tag-Findung kombiniert:
1.  Named Entity Recognition (NER) zur Extraktion von Personen, Orten und Organisationen.
2.  Keyword-Extraktion durch Zählung der häufigsten Substantive (in ihrer Grundform).

Args:
    text (str): Der vollständige Text des Transkripts, der analysiert werden soll.

Returns:
    list: Eine alphabetisch sortierte Liste der gefundenen, einzigartigen Tags.
```

6.2.2 Variable Documentation

6.2.2.1 generated_tags

```
generate_tags.generated_tags = generate_tags(input_text)
```

6.2.2.2 input_text

```
generate_tags.input_text = sys.stdin.read()
```

6.3 run_asr Namespace Reference

Functions

- assign_speakers (transcript_segments, diarization)
- main ()

6.3.1 Function Documentation

6.3.1.1 assign_speakers()

```
run_asr.assign_speakers (
              transcript_segments,
              diarization )
```

```
Ordnet Transkript-Segmenten die erkannten Sprecher zu.

Die Funktion iteriert durch die von Whisper erkannten Textsegmente und gleicht
deren Zeitstempel mit den von pyannote erkannten Sprecher-Zeitintervallen
(diarization) ab.

Args:
    transcript_segments (list): Eine Liste von Segmenten aus der Whisper-Transkription.
    diarization (pyannote.core.Annotation): Das Diarisierungs-Ergebnis von pyannote.

Returns:
    list: Eine Liste von Segmenten, bei denen jedes einen "speaker"-Schlüssel enthält.
```

6.3.1.2 main()

```
run_asr.main ( )
```

```
Hauptfunktion des Skripts zur Transkription und Sprecherzuordnung.
```

Chapter 7

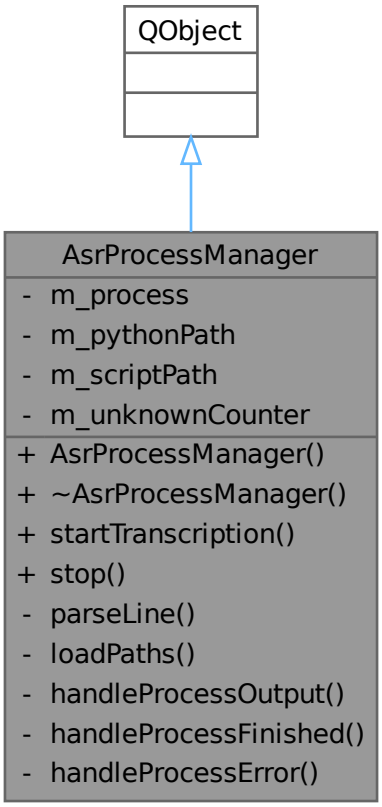
Class Documentation

7.1 AsrProcessManager Class Reference

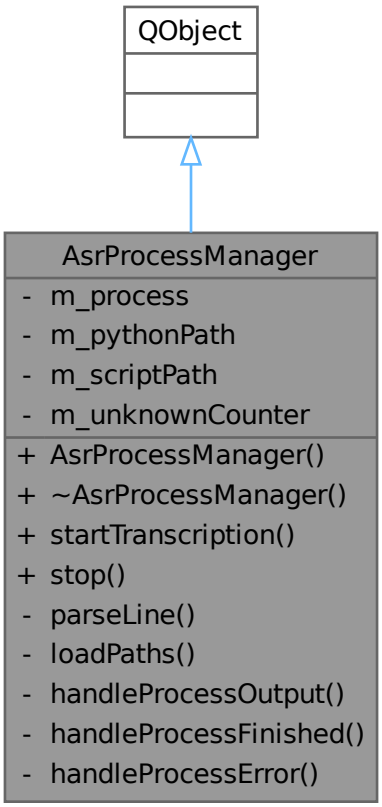
Steuert den externen Python-Prozess für die Spracherkennung (ASR).

`#include <asrprocessmanager.h>`

Inheritance diagram for AsrProcessManager:



Collaboration diagram for AsrProcessManager:



Public Slots

- void startTranscription (const QString &wavFilePath)

  Startet den ASR-Prozess für die angegebene WAV-Datei.
- void stop ()

  Stoppt den laufenden ASR-Prozess, falls einer aktiv ist.

Signals

- void segmentReady (const MetaText &segment)

  Wird für jedes erkannte und geparste Textsegment gesendet.
- void finished (bool success, const QString &errorMsg)

  Wird gesendet, wenn der ASR-Prozess (erfolgreich oder nicht) abgeschlossen ist.

Public Member Functions

- AsrProcessManager (QObject ∗parent=nullptr)

  Konstruktor. Lädt die notwendigen Pfade aus den QSettings.
- ∼AsrProcessManager ()

  Destruktor. Stellt sicher, dass ein laufender Prozess beendet wird.

Private Slots

- void handleProcessOutput ()

  Interner Slot, der aufgerufen wird, wenn der Prozess Daten auf stdout ausgibt.
- void handleProcessFinished (int exitCode, QProcess::ExitStatus exitStatus)

  Interner Slot, der aufgerufen wird, wenn der Prozess sich beendet.
- void handleProcessError (QProcess::ProcessError error)

  Interner Slot, der aufgerufen wird, wenn beim Starten des Prozesses ein Fehler auftritt.

Private Member Functions

- MetaText parseLine (const QString &line)

  Parst eine einzelne Ausgabezeile des Python-Skripts in ein MetaText-Objekt.
- void loadPaths ()

  Lädt den Python- und den Skript-Pfad aus den globalen Einstellungen.

Private Attributes

- QProcess * m_process

  Zeiger auf das QProcess-Objekt, das das Python-Skript ausführt.
- QString m_pythonPath

  Pfad zum Python-Interpreter der virtuellen Umgebung.
- QString m_scriptPath

  Pfad zum ASR-Python-Skript.
- int m_unknownCounter

  Zähler für die Benennung von unbekannten Sprechern (UNKNOWN_0, UNKNOWN_1, ...).

7.1.1 Detailed Description

Steuert den externen Python-Prozess für die Spracherkennung (ASR).

Diese Klasse kapselt die gesamte Logik für das Starten des ASR-Skripts, die Kommunikation über Standard-I/O und die Fehlerbehandlung. Sie arbeitet vollständig asynchron und kommuniziert ihren Zustand über Signale mit dem Rest der Anwendung (z.B. dem MainWindow).

7.1.2 Constructor & Destructor Documentation

7.1.2.1 AsrProcessManager()

```
AsrProcessManager::AsrProcessManager (
            QObject * parent = nullptr )  [explicit]
```

Konstruktor. Lädt die notwendigen Pfade aus den QSettings.

Parameters

parent	Das QObject-Elternteil für die automatische Speicherverwaltung.

7.1.2.2 ~AsrProcessManager()

```
AsrProcessManager::~AsrProcessManager ( )
```

Destruktor. Stellt sicher, dass ein laufender Prozess beendet wird.

7.1.3 Member Function Documentation

7.1.3.1 finished

```
void AsrProcessManager::finished (
            bool success,
            const QString & errorMsg )  [signal]
```

Wird gesendet, wenn der ASR-Prozess (erfolgreich oder nicht) abgeschlossen ist.

Parameters

success	true, wenn der Prozess ohne Fehler (Exit-Code 0) beendet wurde.
errorMsg	Eine Fehlermeldung, falls der Prozess fehlschlug. Enthält typischerweise den Standard-Error-Stream des Prozesses für Debugging.

7.1.3.2 handleProcessError

```
void AsrProcessManager::handleProcessError (
            QProcess::ProcessError error )  [private], [slot]
```

Interner Slot, der aufgerufen wird, wenn beim Starten des Prozesses ein Fehler auftritt.

7.1.3.3 handleProcessFinished

```
void AsrProcessManager::handleProcessFinished (
            int exitCode,
            QProcess::ExitStatus exitStatus )  [private], [slot]
```

Interner Slot, der aufgerufen wird, wenn der Prozess sich beendet.

7.1.3.4 handleProcessOutput

```
void AsrProcessManager::handleProcessOutput ( )  [private], [slot]
```

Interner Slot, der aufgerufen wird, wenn der Prozess Daten auf stdout ausgibt.

7.1.3.5 loadPaths()

```
void AsrProcessManager::loadPaths ( )  [private]
```

Lädt den Python- und den Skript-Pfad aus den globalen Einstellungen.

7.1.3.6 parseLine()

```
MetaText AsrProcessManager::parseLine (
            const QString & line )  [private]
```

Parst eine einzelne Ausgabezeile des Python-Skripts in ein MetaText-Objekt.

Parameters

line	Die zu parsende Zeile im Format "[start]s --> [end]s] SPEAKER: Text".

Returns

Ein gefülltes MetaText-Objekt. Wenn das Parsen fehlschlägt, ist das Objekt leer.

7.1.3.7 segmentReady

```
void AsrProcessManager::segmentReady (
            const MetaText & segment )  [signal]
```

Wird für jedes erkannte und geparste Textsegment gesendet.

Das Python-Skript gibt die Segmente zeilenweise aus; dieses Signal wird für jede erfolgreich geparste Zeile emittiert.

Parameters

segment	Das vollständig geparste Segment mit Zeitstempeln, Sprecher und Text.

7.1.3.8 startTranscription

```
void AsrProcessManager::startTranscription (
            const QString & wavFilePath )  [slot]
```

Startet den ASR-Prozess für die angegebene WAV-Datei.

Stellt sicher, dass nicht bereits ein Prozess läuft und übergibt den Dateipfad als Argument an das Python-Skript.

Parameters

wavFilePath	Der absolute Pfad zur 16-kHz-Mono-WAV-Datei, die verarbeitet werden soll.

Note

Dies ist ein öffentlicher Slot, der z.B. von der MainWindow aufgerufen wird.

7.1.3.9 stop

```
void AsrProcessManager::stop ( )  [slot]
```

Stoppt den laufenden ASR-Prozess, falls einer aktiv ist.

Nützlich, wenn eine neue Aufnahme gestartet wird, während eine alte Transkription noch läuft.

7.1.4 Member Data Documentation

7.1.4.1 m_process

```
QProcess* AsrProcessManager::m_process  [private]
```

Zeiger auf das QProcess-Objekt, das das Python-Skript ausführt.

7.1.4.2 m_pythonPath

```
QString AsrProcessManager::m_pythonPath  [private]
```

Pfad zum Python-Interpreter der virtuellen Umgebung.

7.1.4.3 m_scriptPath

```
QString AsrProcessManager::m_scriptPath  [private]
```

Pfad zum ASR-Python-Skript.

7.1.4.4 m_unknownCounter

```
int AsrProcessManager::m_unknownCounter  [private]
```

Zähler für die Benennung von unbekannten Sprechern (UNKNOWN_0, UNKNOWN_1, ...).

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

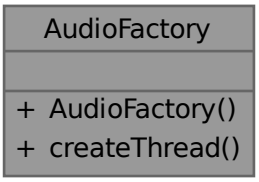
- asrprocessmanager.h
- asrprocessmanager.cpp

7.2 AudioFactory Class Reference

Eine Factory-Klasse zur Erstellung von plattformspezifischen CaptureThread-Objekten.

`#include <audiofactory.h>`

Collaboration diagram for AudioFactory:



Public Member Functions

- AudioFactory ()=default
  Standard-Konstruktor.

Static Public Member Functions

- static CaptureThread * createThread (QObject *parent=nullptr)
  Erstellt eine Instanz der korrekten, plattformspezifischen CaptureThread-Klasse.

7.2.1 Detailed Description

Eine Factory-Klasse zur Erstellung von plattformspezifischen CaptureThread-Objekten.

Diese Klasse nutzt den Factory-Pattern, um die Logik zur Auswahl der korrekten CaptureThread-Implementierung (z.B. PulseCaptureThread für Linux, WinCaptureThread für Windows) von der restlichen Anwendung zu entkoppeln.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 AudioFactory()

```
AudioFactory::AudioFactory ( )  [default]
```

Standard-Konstruktor.

7.2.3 Member Function Documentation

7.2.3.1 createThread()

```
CaptureThread * AudioFactory::createThread (
            QObject * parent = nullptr )  [static]
```

Erstellt eine Instanz der korrekten, plattformspezifischen CaptureThread-Klasse.

Diese Methode verwendet Präprozessor-Direktiven, um zur Kompilierzeit die passende Thread-Implementierung für das Zielbetriebssystem auszuwählen.

Parameters

parent	Das QObject-Elternteil für den neuen Thread, um die Speicherverwaltung zu gewährleisten.

Returns

Ein Zeiger auf den neu erstellten CaptureThread oder nullptr, wenn die Plattform nicht unterstützt wird.

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

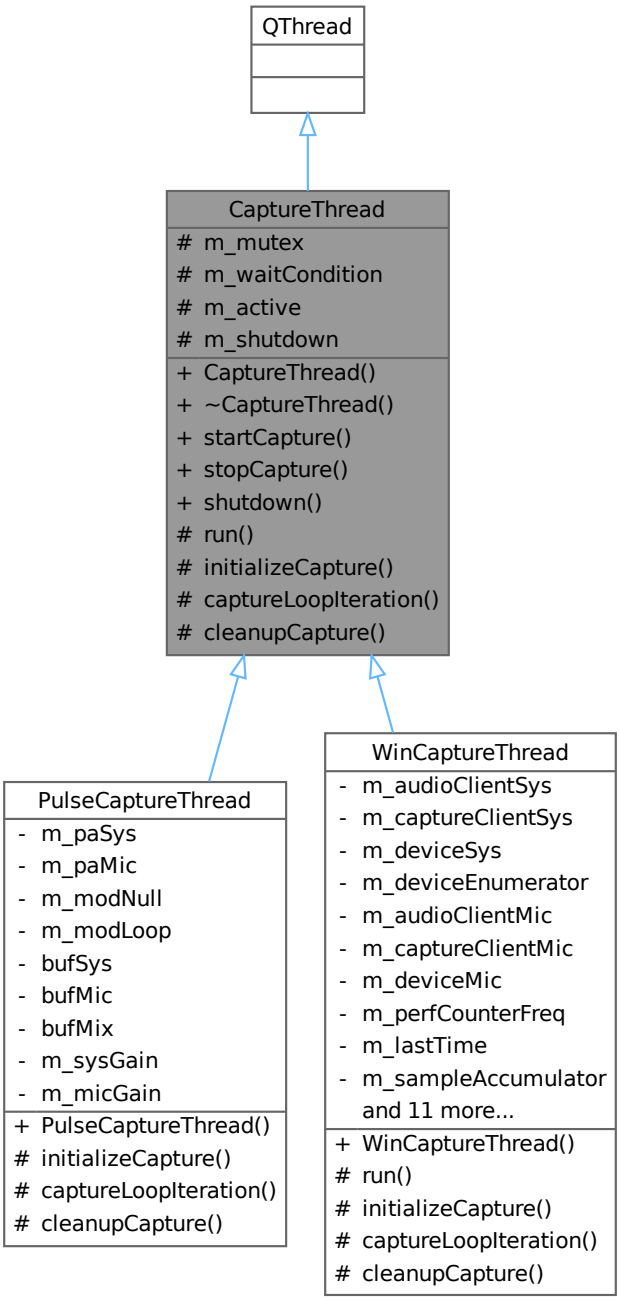
- audiofactory.h
- audiofactory.cpp

7.3 CaptureThread Class Reference

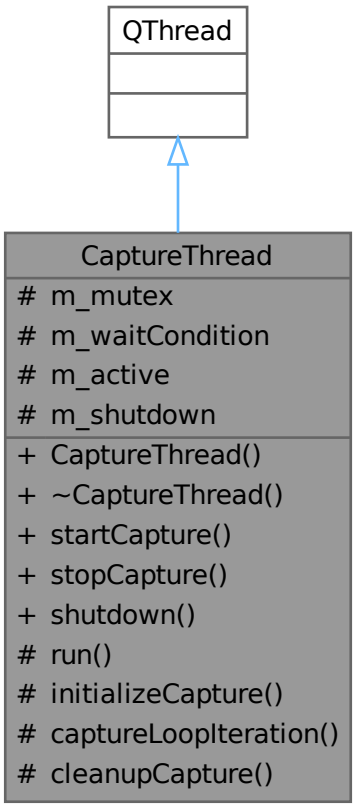
Eine abstrakte Basisklasse für Threads, die Audio in Echtzeit aufnehmen.

`#include <capturethread.h>`

Inheritance diagram for CaptureThread:



Collaboration diagram for CaptureThread:



Signals

- void pcmChunkReady (QList< float > chunk)

  Wird gesendet, wenn ein neuer Block von Audio-Daten (PCM-Samples) bereitsteht.
- void started ()

  Wird gesendet, unmittelbar nachdem die plattformspezifische Initialisierung erfolgreich war und die Aufnahmeschleife beginnt.
- void stopped ()

  Wird gesendet, nachdem die Aufnahmeschleife beendet und die Aufräumarbeiten abgeschlossen sind.

Public Member Functions

- CaptureThread (QObject *parent=nullptr)

  Standard-Konstruktor.
- virtual ~CaptureThread ()=default

  Virtueller Destruktor.
- void startCapture ()

  Startet eine neue Aufnahme-Session. Diese Methode ist thread-sicher und weckt den run()-Loop auf, um mit der Aufnahme zu beginnen.

- virtual void stopCapture ()

  Fordert das Beenden der aktuellen Aufnahme-Session an. Dies ist ein nicht-blockierender Aufruf. Der Thread beendet die Aufnahmeschleife so bald wie möglich.
- void shutdown ()

  Beendet den Thread vollständig und wartet auf dessen Terminierung. Dies ist ein blockierender Aufruf, der sicherstellt, dass alle Ressourcen freigegeben werden.

Protected Member Functions

- void run () override

  Die Hauptfunktion des Threads, die von QThread aufgerufen wird. Sie implementiert die Zustandsmaschine für den Aufnahme-Lebenszyklus.
- virtual bool initializeCapture ()=0

  Rein virtuelle Methode zur Initialisierung der plattformspezifischen Audio-Ressourcen. Muss von abgeleiteten Klassen implementiert werden.
- virtual void captureLoopIteration ()=0

  Rein virtuelle Methode, die eine einzelne Iteration der Aufnahmeschleife durchführt. Hier werden die Audiodaten vom Gerät gelesen und verarbeitet. Muss von abgeleiteten Klassen implementiert werden.
- virtual void cleanupCapture ()=0

  Rein virtuelle Methode zum Aufräumen und Freigeben der plattformspezifischen Ressourcen. Wird aufgerufen, nachdem die Aufnahmeschleife beendet wurde. Muss von abgeleiteten Klassen implementiert werden.

Protected Attributes

- QMutex m_mutex

  Schützt den Zugriff auf den Zustand des Threads.
- QWaitCondition m_waitCondition

  Lässt den Thread schlafen, wenn er inaktiv ist.
- std::atomic< bool > m_active

  Steuert die innere Aufnahmeschleife (start/stop).
- std::atomic< bool > m_shutdown

  Signalisiert dem Thread, sich komplett zu beenden.

7.3.1 Detailed Description

Eine abstrakte Basisklasse für Threads, die Audio in Echtzeit aufnehmen.

Diese Klasse implementiert die allgemeine Logik und den Lebenszyklus eines Audio-Aufnahme-Threads mithilfe des Template-Method-Patterns. Die run()-Methode definiert das Grundgerüst des Ablaufs, während plattformspezifische Details (Initialisierung, die eigentliche Aufnahmeschleife und das Aufräumen) von abgeleiteten Klassen implementiert werden müssen.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 CaptureThread()

```
CaptureThread::CaptureThread (
            QObject * parent = nullptr )  [explicit]
```

Standard-Konstruktor.

Parameters

parent	Das QObject-Elternteil für die Speicherverwaltung.

7.3.2.2 ~CaptureThread()

```
virtual CaptureThread::~CaptureThread ( )  [virtual], [default]
```

Virtueller Destruktor.

7.3.3 Member Function Documentation

7.3.3.1 captureLoopIteration()

```
virtual void CaptureThread::captureLoopIteration ( )  [protected], [pure virtual]
```

Rein virtuelle Methode, die eine einzelne Iteration der Aufnahmeschleife durchführt. Hier werden die Audiodaten vom Gerät gelesen und verarbeitet. Muss von abgeleiteten Klassen implementiert werden.

Implemented in PulseCaptureThread, and WinCaptureThread.

7.3.3.2 cleanupCapture()

```
virtual void CaptureThread::cleanupCapture ( )  [protected], [pure virtual]
```

Rein virtuelle Methode zum Aufräumen und Freigeben der plattformspezifischen Ressourcen. Wird aufgerufen, nachdem die Aufnahmeschleife beendet wurde. Muss von abgeleiteten Klassen implementiert werden.

Implemented in PulseCaptureThread, and WinCaptureThread.

7.3.3.3 initializeCapture()

```
virtual bool CaptureThread::initializeCapture ( )  [protected], [pure virtual]
```

Rein virtuelle Methode zur Initialisierung der plattformspezifischen Audio-Ressourcen. Muss von abgeleiteten Klassen implementiert werden.

Returns

true bei Erfolg, andernfalls false.

Implemented in PulseCaptureThread, and WinCaptureThread.

7.3.3.4 pcmChunkReady

```
void CaptureThread::pcmChunkReady (
            QList< float > chunk )  [signal]
```

Wird gesendet, wenn ein neuer Block von Audio-Daten (PCM-Samples) bereitsteht.

Parameters

chunk	Eine Liste von float-Werten, die die Audio-Samples repräsentieren.

7.3.3.5 run()

```
void CaptureThread::run ( )  [override], [protected]
```

Die Hauptfunktion des Threads, die von QThread aufgerufen wird. Sie implementiert die Zustandsmaschine für den Aufnahme-Lebenszyklus.

7.3.3.6 shutdown()

```
void CaptureThread::shutdown ( )
```

Beendet den Thread vollständig und wartet auf dessen Terminierung. Dies ist ein blockierender Aufruf, der sicherstellt, dass alle Ressourcen freigegeben werden.

7.3.3.7 startCapture()

```
void CaptureThread::startCapture ( )
```

Startet eine neue Aufnahme-Session. Diese Methode ist thread-sicher und weckt den run()-Loop auf, um mit der Aufnahme zu beginnen.

7.3.3.8 started

```
void CaptureThread::started ( )  [signal]
```

Wird gesendet, unmittelbar nachdem die plattformspezifische Initialisierung erfolgreich war und die Aufnahmeschleife beginnt.

7.3.3.9 stopCapture()

```
void CaptureThread::stopCapture ( )  [virtual]
```

Fordert das Beenden der aktuellen Aufnahme-Session an. Dies ist ein nicht-blockierender Aufruf. Der Thread beendet die Aufnahmeschleife so bald wie möglich.

7.3.3.10 stopped

```
void CaptureThread::stopped ( )  [signal]
```

Wird gesendet, nachdem die Aufnahmeschleife beendet und die Aufräumarbeiten abgeschlossen sind.

7.3.4 Member Data Documentation

7.3.4.1 m_active

```
std::atomic<bool> CaptureThread::m_active  [protected]
```

Steuert die innere Aufnahmeschleife (start/stop).

7.3.4.2 m_mutex

```
QMutex CaptureThread::m_mutex  [protected]
```

Schützt den Zugriff auf den Zustand des Threads.

7.3.4.3 m_shutdown

```
std::atomic<bool> CaptureThread::m_shutdown  [protected]
```

Signalisiert dem Thread, sich komplett zu beenden.

7.3.4.4 m_waitCondition

```
QWaitCondition CaptureThread::m_waitCondition  [protected]
```

Lässt den Thread schlafen, wenn er inaktiv ist.

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

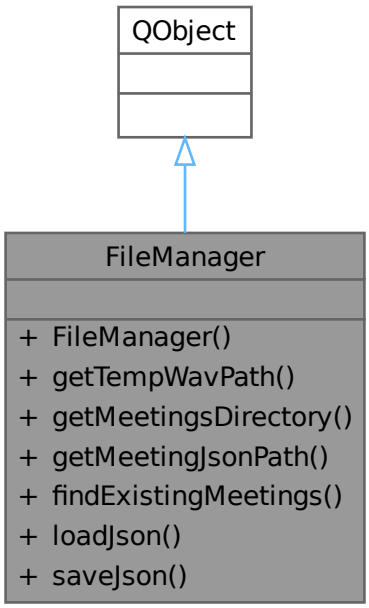
- capturethread.h
- capturethread.cpp

7.4 FileManager Class Reference

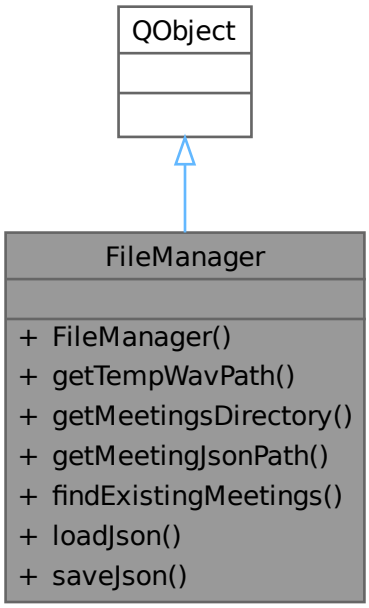
Kapselt alle direkten Dateisystem-Interaktionen der Anwendung.

```
#include <filemanager.h>
```

Inheritance diagram for FileManager:



Collaboration diagram for FileManager:



Public Member Functions

- FileManager (QObject ∗parent=nullptr)

  Konstruktor, der das Meeting-Verzeichnis initialisiert.
- QString getTempWavPath (bool forAsr=false) const

  Gibt den vollständigen Pfad für die temporäre WAV-Aufnahmedatei zurück.
- QString getMeetingsDirectory () const

  Gibt den Pfad zum Verzeichnis zurück, in dem alle Meeting-Transkripte gespeichert werden.
- QString getMeetingJsonPath (const QString &meetingId, bool isEdited) const

  Erstellt den vollständigen Dateipfad für eine bestimmte Meeting-JSON-Datei.
- QStringList findExistingMeetings () const

  Durchsucht das Meeting-Verzeichnis und gibt eine Liste aller gefundenen Meeting-IDs zurück.
- QJsonDocument loadJson (const QString &filePath, bool &ok) const

  Lädt eine JSON-Datei vom angegebenen Pfad.
- bool saveJson (const QString &filePath, const QJsonDocument &doc) const

  Speichert ein QJsonDocument in der angegebenen Datei.

7.4.1 Detailed Description

Kapselt alle direkten Dateisystem-Interaktionen der Anwendung.

Diese Klasse dient als zentrale Anlaufstelle ("Single Source of Truth") für alle Pfad-Konstruktionen und Dateioperationen. Sie abstrahiert Details wie temporäre Verzeichnisse oder die Namenskonvention von Meeting-Dateien vom Rest der Anwendung. Sie erbt von QObject, um die automatische Speicherverwaltung durch Qt zu nutzen.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 FileManager()

```
FileManager::FileManager (
            QObject * parent = nullptr )  [explicit]
```

Konstruktor, der das Meeting-Verzeichnis initialisiert.

Parameters

parent	Das QObject-Elternteil für die Speicherverwaltung.

7.4.3 Member Function Documentation

7.4.3.1 findExistingMeetings()

```
QStringList FileManager::findExistingMeetings ( ) const
```

Durchsucht das Meeting-Verzeichnis und gibt eine Liste aller gefundenen Meeting-IDs zurück.

Todo Diese Methode ist nur ein temporärer Platzhalter für die Demonstration und wird entfernt, sobald die Anwendung mit der Datenbank verknüpft wird.

Returns

Eine QStringList mit den Basisnamen der Meetings (ohne Suffix).

7.4.3.2 getMeetingJsonPath()

```
QString FileManager::getMeetingJsonPath (
            const QString & meetingId,
            bool isEdited ) const
```

Erstellt den vollständigen Dateipfad für eine bestimmte Meeting-JSON-Datei.

Todo Diese Methode ist nur ein temporärer Platzhalter für die Demonstration und wird entfernt, sobald die Anwendung mit der Datenbank verknüpft wird.

Parameters

meeting↩ Id	Die eindeutige ID des Meetings (z.B. "Aufnahme - 2025-06-17_09-30").
isEdited	Wenn true, wird der Pfad zur "_bearbeitet.json"-Version zurückgegeben, andernfalls zur "_original.json".

Returns

Der vollständige Dateipfad.

7.4.3.3 getMeetingsDirectory()

```
QString FileManager::getMeetingsDirectory ( ) const
```

Gibt den Pfad zum Verzeichnis zurück, in dem alle Meeting-Transkripte gespeichert werden.

Note

Stellt bei der ersten Ausführung sicher, dass das Verzeichnis existiert.

Todo Diese Methode ist nur ein temporärer Platzhalter für die Demonstration und wird entfernt, sobald die Anwendung mit der Datenbank verknüpft wird.

Returns

Der vollständige Verzeichnispfad.

7.4.3.4 getTempWavPath()

```
QString FileManager::getTempWavPath (
            bool forAsr = false ) const
```

Gibt den vollständigen Pfad für die temporäre WAV-Aufnahmedatei zurück.

Parameters

forAsr	Wenn true, wird der Pfad für die heruntergesampelte ASR-Version zurückgegeben.

Returns

Der vollständige Dateipfad.

7.4.3.5 loadJson()

```
QJsonDocument FileManager::loadJson (
            const QString & filePath,
            bool & ok ) const
```

Lädt eine JSON-Datei vom angegebenen Pfad.

Todo Diese Methode ist nur ein temporärer Platzhalter für die Demonstration bis die Anwendung mit der Datenbank verknüpft wird. Danach kann sie aber evtl. erhalten bleiben.

Parameters

filePath	Der Pfad zur zu ladenden Datei.
ok	[out] Wird auf true gesetzt, wenn das Laden und Parsen erfolgreich war, sonst false.

Returns

Das geladene QJsonDocument. Bei einem Fehler ist das Dokument leer.

7.4.3.6 saveJson()

```
bool FileManager::saveJson (
            const QString & filePath,
            const QJsonDocument & doc ) const
```

Speichert ein QJsonDocument in der angegebenen Datei.

Todo Diese Methode ist nur ein temporärer Platzhalter für die Demonstration bis die Anwendung mit der Datenbank verknüpft wird. Danach kann sie aber evtl. erhalten bleiben.

Parameters

filePath	Der Ziel-Dateipfad.
doc	Das zu speichernde JSON-Dokument.

Returns

true bei Erfolg, andernfalls false.

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

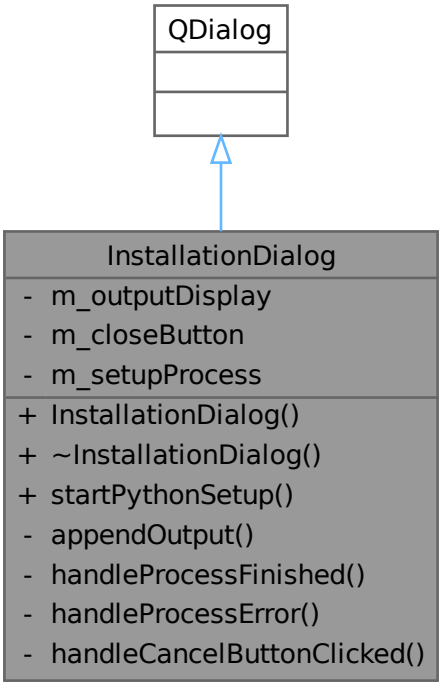
- filemanager.h
- filemanager.cpp

7.5 InstallationDialog Class Reference

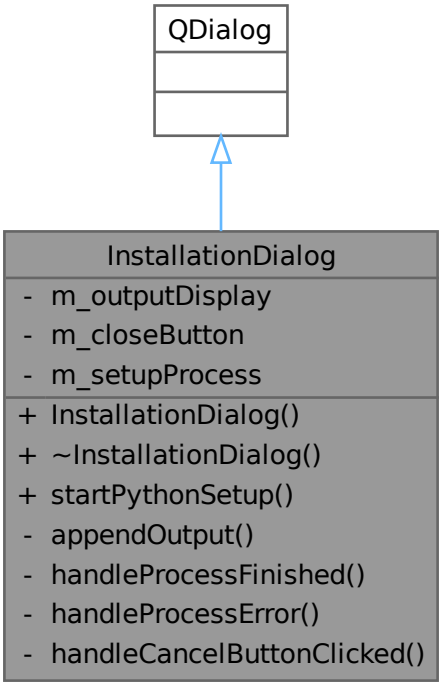
Ein modaler Dialog, der die Ausgabe des Python-Setup-Skripts anzeigt.

`#include <installationdialog.h>`

Inheritance diagram for InstallationDialog:



Collaboration diagram for InstallationDialog:



Public Slots

- void startPythonSetup ()

  Startet die Ausführung des plattformspezifischen Python-Setup-Skripts.

Signals

- void installationFinished (bool success, const QString &errorMessage="")

  Wird emittiert, wenn der Installationsprozess beendet ist.

Public Member Functions

- InstallationDialog (QWidget *parent=nullptr)
- ~InstallationDialog ()

Private Slots

- void appendOutput ()

  Hängt die Ausgaben (stdout/stderr) des Prozesses an das Textfeld an.
- void handleProcessFinished (int exitCode, QProcess::ExitStatus exitStatus)

  Behandelt das `finished`-Signal des QProcess.
- void handleProcessError (QProcess::ProcessError error)

  Behandelt das `errorOccurred`-Signal des QProcess.
- void handleCancelButtonClicked ()

  Behandelt den Klick auf den Abbrechen/Schließen-Button.

Private Attributes

- QTextEdit * m_outputDisplay

  Zeigt die Konsolenausgabe des Setup-Skripts an.
- QPushButton * m_closeButton

  Button zum Abbrechen oder Schließen des Dialogs.
- QProcess * m_setupProcess

  Der Prozess, der das Setup-Skript ausführt.

7.5.1 Detailed Description

Ein modaler Dialog, der die Ausgabe des Python-Setup-Skripts anzeigt.

Dieser Dialog startet das Setup-Skript in einem eigenen QProcess und leitet dessen Standard- und Fehlerausgabe in ein Textfeld um. Er meldet das Ergebnis des Prozesses über ein Signal zurück.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 InstallationDialog()

```
InstallationDialog::InstallationDialog (
            QWidget * parent = nullptr )  [explicit]
```

7.5.2.2 ~InstallationDialog()

```
InstallationDialog::~InstallationDialog ( )
```

7.5.3 Member Function Documentation

7.5.3.1 appendOutput

```
void InstallationDialog::appendOutput ( )  [private], [slot]
```

Hängt die Ausgaben (stdout/stderr) des Prozesses an das Textfeld an.

7.5.3.2 handleCancelButtonClicked

```
void InstallationDialog::handleCancelButtonClicked ( )  [private], [slot]
```

Behandelt den Klick auf den Abbrechen/Schließen-Button.

7.5.3.3 handleProcessError

```
void InstallationDialog::handleProcessError (
            QProcess::ProcessError error )  [private], [slot]
```

Behandelt das `errorOccurred`-Signal des QProcess.

7.5.3.4 handleProcessFinished

```
void InstallationDialog::handleProcessFinished (
            int exitCode,
            QProcess::ExitStatus exitStatus )  [private], [slot]
```

Behandelt das `finished`-Signal des QProcess.

7.5.3.5 installationFinished

```
void InstallationDialog::installationFinished (
            bool success,
            const QString & errorMessage = "" )  [signal]
```

Wird emittiert, wenn der Installationsprozess beendet ist.

Parameters

success	`true`, wenn der Prozess erfolgreich war.
errorMessage	Eine Fehlermeldung bei Misserfolg.

7.5.3.6 startPythonSetup

```
void InstallationDialog::startPythonSetup ( )  [slot]
```

Startet die Ausführung des plattformspezifischen Python-Setup-Skripts.

7.5.4 Member Data Documentation

7.5.4.1 m_closeButton

```
QPushButton* InstallationDialog::m_closeButton  [private]
```

Button zum Abbrechen oder Schließen des Dialogs.

7.5.4.2 m_outputDisplay

```
QTextEdit* InstallationDialog::m_outputDisplay  [private]
```

Zeigt die Konsolenausgabe des Setup-Skripts an.

7.5.4.3 m_setupProcess

```
QProcess* InstallationDialog::m_setupProcess  [private]
```

Der Prozess, der das Setup-Skript ausführt.

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

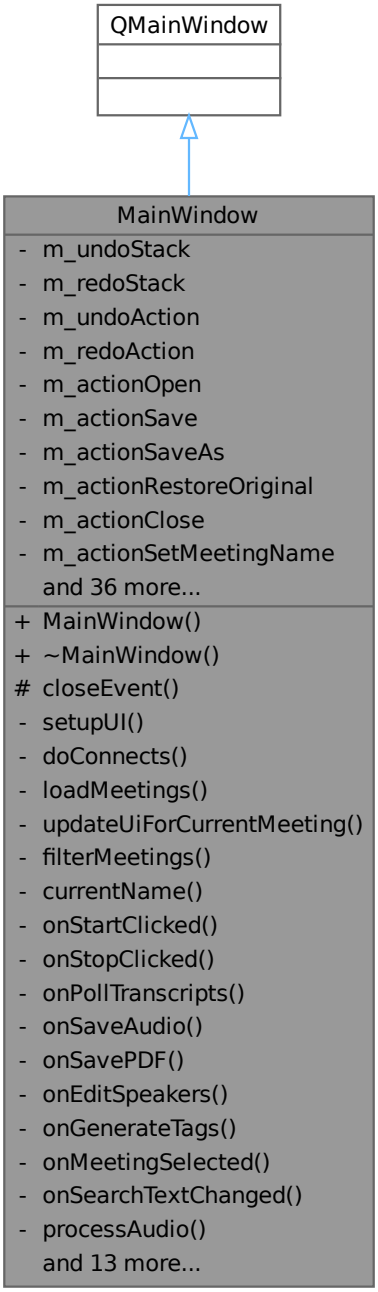
- installationdialog.h
- installationdialog.cpp

7.6 MainWindow Class Reference

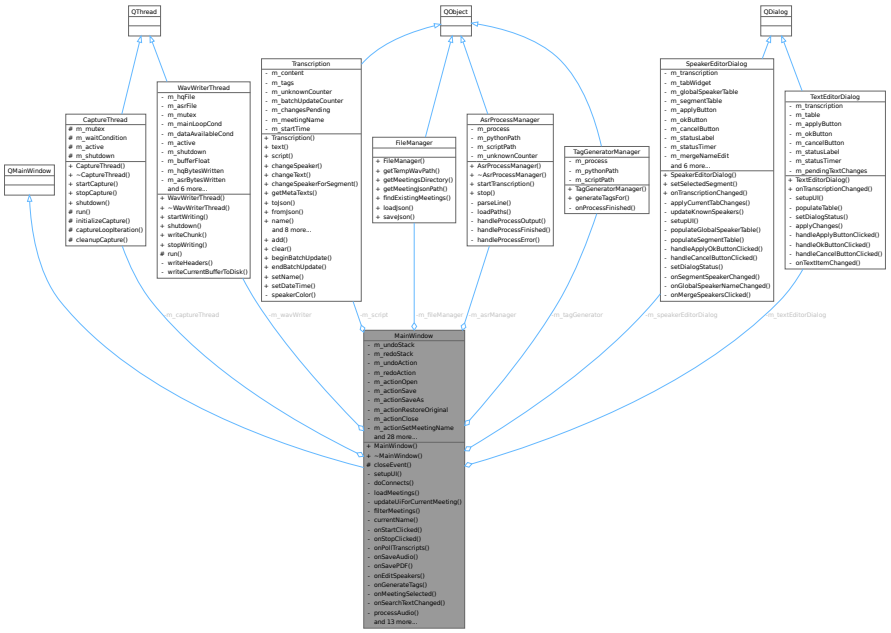
Das Hauptfenster und die zentrale Steuerungseinheit der Anwendung.

`#include <mainwindow.h>`

Inheritance diagram for MainWindow:



Collaboration diagram for MainWindow:



Public Member Functions

- MainWindow (QWidget ∗parent=nullptr)
- ∼MainWindow () override

Protected Member Functions

- void closeEvent (QCloseEvent ∗event) override

  Wird aufgerufen, wenn der Benutzer versucht, das Fenster zu schließen.

Private Slots

- void onStartClicked ()

  Startet eine neue Audio-Aufnahmesession.
- void onStopClicked ()

  Beendet die laufende Audio-Aufnahmesession.
- void onPollTranscripts ()

  Temporärer Platzhalter zur Simulation von Echtzeit-Transkripten.
- void onSaveAudio ()

  Öffnet einen Dialog zum Speichern der hochqualitativen WAV-Audiodatei.
- void onSavePDF ()

  Delegiert die Erstellung und Anzeige einer PDF-Version des Transkripts.
- void onEditSpeakers ()

  Öffnet den Dialog zur Bearbeitung und Zuordnung von Sprechern.
- void onGenerateTags ()

  Startet den Prozess zur automatischen Generierung von Tags für das Transkript.
- void onMeetingSelected (QListWidgetItem ∗item)

Lädt das ausgewählte Meeting aus der Liste bei einem Doppelklick.

- void onSearchTextChanged (const QString &text)

  Filtert die Meeting-Liste basierend auf der Eingabe im Suchfeld.
- void processAudio ()

  Startet den ASR-Prozess für die zuletzt aufgenommene Audiodatei.
- void openSettingsWizard ()

  Öffnet den Einstellungsdialog.
- void setStatus (const QString &text, bool keep=false)

  Zeigt eine Nachricht in der Statusleiste an.
- void onEditTranscript ()

  Öffnet den Dialog zur Bearbeitung des Transkript-Textes.
- void onUndo ()

  Macht die letzte Änderung am Transkript rückgängig.
- void onRedo ()

  Wiederholt die letzte rückgängig gemachte Änderung.
- void loadTranscriptionFromJson ()

  Öffnet einen Dateidialog, um ein Transkript im JSON-Format zu laden.
- void saveTranscriptionToJson ()

  Speichert das aktuell bearbeitete Transkript.
- void saveTranscriptionToJsonAs ()

  Öffnet einen Dateidialog, um das Transkript unter einem neuen Namen zu speichern.
- void restoreOriginalTranscription ()

  Setzt das aktuelle Transkript auf die ursprünglich generierte Version zurück.
- void updateUndoRedoState ()

  Aktualisiert den Zustand der Undo/Redo-Buttons.
- void setMeetingName (const QString &name)

  Setzt den Namen für das aktuelle Meeting.
- void onSetMeetingName ()

  Öffnet einen Dialog, um den Meeting-Namen manuell einzugeben.
- void onReinstallPython ()

  Startet den Prozess zur Neuinstallation der Python-Umgebung.

Private Member Functions

- void setupUI ()

  Erstellt und arrangiert alle UI-Widgets.
- void doConnects ()

  Bündelt alle Signal-Slot-Verbindungen.
- void loadMeetings ()

  Lädt die Liste der verfügbaren Meetings in die Seitenleiste.
- void updateUiForCurrentMeeting ()

  Aktualisiert den Zustand der UI (Buttons, Labels) basierend auf dem aktuellen Meeting-Status.
- void filterMeetings (const QString &filter)

  Wendet einen Filter auf die sichtbaren Elemente der Meeting-Liste an.
- QString currentName () const

  Konstruiert den Anzeige-Namen für das aktuelle Meeting aus Name und Datum.

Private Attributes

- QStack< QJsonDocument > m_undoStack

  Stapel für die Undo-Zustände.
- QStack< QJsonDocument > m_redoStack

  Stapel für die Redo-Zustände.
- QAction * m_undoAction

  Menü-Aktion für Undo.
- QAction * m_redoAction

  Menü-Aktion für Redo.
- QAction * m_actionOpen
- QAction * m_actionSave
- QAction * m_actionSaveAs
- QAction * m_actionRestoreOriginal
- QAction * m_actionClose
- QAction * m_actionSetMeetingName
- QAction * m_settingsAction
- QAction * m_reinstallPythonAction
- CaptureThread * m_captureThread

  Thread für die plattformspezifische Audio-Aufnahme.
- WavWriterThread * m_wavWriter

  Thread zum Schreiben der WAV-Dateien.
- Transcription * m_script

  Das zentrale Datenmodell für das Transkript.
- FileManager * m_fileManager

  Manager für alle Dateizugriffe.
- AsrProcessManager * m_asrManager

  Manager für den ASR-Python-Prozess.
- TagGeneratorManager * m_tagGenerator

  Manager für den Tag-Generator-Python-Prozess.
- QSplitter * splitter
- QWidget * leftPanel
- QLineEdit * searchBox
- QListWidget * meetingList
- QWidget * rightPanel
- QVBoxLayout * mainLayout
- QHBoxLayout * buttonLayout
- QPushButton * startButton
- QPushButton * stopButton
- QPushButton * saveAudioButton
- QPushButton * savePDFButton
- QPushButton * assignNamesButton
- QPushButton * generateTagsButton
- QPushButton * editTextButton
- QLabel * timeLabel
- QLabel * nameLabel
- QLabel * statusLabel
- QTextEdit * transcriptView
- QTimer * pollTimer
- QTimer * timeUpdateTimer
- QTimer * statusTimer
- QElapsedTimer elapsedTime
- SpeakerEditorDialog * m_speakerEditorDialog

- TextEditorDialog * m_textEditorDialog
- QString m_currentAudioPath

  Pfad zur zuletzt gespeicherten Audiodatei.
- QString m_currentMeetingName

  Name des aktuellen Meetings (wird bei Aufnahme/Laden gesetzt).
- QString m_currentMeetingDateTime

  Zeitstempel des aktuellen Meetings.
- QProcess * pluginProcess

  Platzhalter für einen möglichen IPC-Prozess.

7.6.1 Detailed Description

Das Hauptfenster und die zentrale Steuerungseinheit der Anwendung.

Die MainWindow-Klasse ist verantwortlich für die Darstellung der Benutzeroberfläche und die Koordination der verschiedenen Hintergrundprozesse und Manager-Klassen. Sie nimmt Benutzerinteraktionen entgegen und delegiert die Aufgaben an die zuständigen Komponenten.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 MainWindow()

```
MainWindow::MainWindow (
            QWidget * parent = nullptr )  [explicit]
```

7.6.2.2 ~MainWindow()

```
MainWindow::~MainWindow ( )  [override]
```

7.6.3 Member Function Documentation

7.6.3.1 closeEvent()

```
void MainWindow::closeEvent (
            QCloseEvent * event )  [override], [protected]
```

Wird aufgerufen, wenn der Benutzer versucht, das Fenster zu schließen.

Parameters

event	Das Close-Event, das ggf. ignoriert werden kann, um das Schließen zu verhindern.

Todo Die Logik zum Speichern bei ungesicherten Änderungen soll zukünftig ebenfalls die **Datenbank-Speicherfunktion** aufrufen.

7.6.3.2 currentName()

```
QString MainWindow::currentName ( ) const  [private]
```

Konstruiert den Anzeige-Namen für das aktuelle Meeting aus Name und Datum.

7.6.3.3 doConnects()

```
void MainWindow::doConnects ( )  [private]
```

Bündelt alle Signal-Slot-Verbindungen.

7.6.3.4 filterMeetings()

```
void MainWindow::filterMeetings (
            const QString & filter )  [private]
```

Wendet einen Filter auf die sichtbaren Elemente der Meeting-Liste an.

7.6.3.5 loadMeetings()

```
void MainWindow::loadMeetings ( )  [private]
```

Lädt die Liste der verfügbaren Meetings in die Seitenleiste.

Todo Aktuell wird das Dateisystem durchsucht. Zukünftig soll diese Methode eine **Datenbankabfrage** ausführen, um alle gespeicherten Meetings abzurufen.

7.6.3.6 loadTranscriptionFromJson

```
void MainWindow::loadTranscriptionFromJson ( )  [private], [slot]
```

Öffnet einen Dateidialog, um ein Transkript im JSON-Format zu laden.

Todo Diese Funktion zum Laden einer beliebigen Datei bleibt bestehen, aber die primäre Ladefunktion wird der Datenbankzugriff über onMeetingSelected().

7.6.3.7 onEditSpeakers

```
void MainWindow::onEditSpeakers ( )  [private], [slot]
```

Öffnet den Dialog zur Bearbeitung und Zuordnung von Sprechern.

7.6.3.8 onEditTranscript

```
void MainWindow::onEditTranscript ( )  [private], [slot]
```

Öffnet den Dialog zur Bearbeitung des Transkript-Textes.

7.6.3.9 onGenerateTags

```
void MainWindow::onGenerateTags ( )  [private], [slot]
```

Startet den Prozess zur automatischen Generierung von Tags für das Transkript.

7.6.3.10 onMeetingSelected

```
void MainWindow::onMeetingSelected (
            QListWidgetItem * item )  [private], [slot]
```

Lädt das ausgewählte Meeting aus der Liste bei einem Doppelklick.

Parameters

item	Das angeklickte Listenelement, dessen Text die Meeting-ID enthält.

Todo Die Methode verwendet derzeit den FileManager, um eine JSON-Datei zu laden. Dies soll durch eine **Datenbankabfrage** ersetzt werden, die das Meeting anhand seiner ID lädt.

7.6.3.11 onPollTranscripts

```
void MainWindow::onPollTranscripts ( )  [private], [slot]
```

Temporärer Platzhalter zur Simulation von Echtzeit-Transkripten.

Todo Diese Methode ist nur für Demonstrationszwecke und wird entfernt, sobald die Echtzeit-Verarbeitung via ASR-Manager implementiert ist.

7.6.3.12 onRedo

```
void MainWindow::onRedo ( )  [private], [slot]
```

Wiederholt die letzte rückgängig gemachte Änderung.

7.6.3.13 onReinstallPython

```
void MainWindow::onReinstallPython ( )  [private], [slot]
```

Startet den Prozess zur Neuinstallation der Python-Umgebung.

Note

Zeigt einen Bestätigungsdialog vor der Ausführung.

7.6.3.14 onSaveAudio

```
void MainWindow::onSaveAudio ( )  [private], [slot]
```

Öffnet einen Dialog zum Speichern der hochqualitativen WAV-Audiodatei.

7.6.3.15 onSavePDF

```
void MainWindow::onSavePDF ( )  [private], [slot]
```

Delegiert die Erstellung und Anzeige einer PDF-Version des Transkripts.

7.6.3.16 onSearchTextChanged

```
void MainWindow::onSearchTextChanged (
            const QString & text )  [private], [slot]
```

Filtert die Meeting-Liste basierend auf der Eingabe im Suchfeld.

7.6.3.17 onSetMeetingName

```
void MainWindow::onSetMeetingName ( )  [private], [slot]
```

Öffnet einen Dialog, um den Meeting-Namen manuell einzugeben.

7.6.3.18 onStartClicked

```
void MainWindow::onStartClicked ( )  [private], [slot]
```

Startet eine neue Audio-Aufnahmesession.

7.6.3.19 onStopClicked

```
void MainWindow::onStopClicked ( )  [private], [slot]
```

Beendet die laufende Audio-Aufnahmesession.

7.6.3.20 onUndo

```
void MainWindow::onUndo ( )  [private], [slot]
```

Macht die letzte Änderung am Transkript rückgängig.

7.6.3.21 openSettingsWizard

```
void MainWindow::openSettingsWizard ( )  [private], [slot]
```

Öffnet den Einstellungsdialog.

7.6.3.22 processAudio

```
void MainWindow::processAudio ( )  [private], [slot]
```

Startet den ASR-Prozess für die zuletzt aufgenommene Audiodatei.

7.6.3.23 restoreOriginalTranscription

```
void MainWindow::restoreOriginalTranscription ( )  [private], [slot]
```

Setzt das aktuelle Transkript auf die ursprünglich generierte Version zurück.

Todo Lädt aktuell das `_original.json`-File. Zukünftig soll dies den Originalzustand des Transkripts aus der **Datenbank** wiederherstellen.

7.6.3.24 saveTranscriptionToJson

```
void MainWindow::saveTranscriptionToJson ( )  [private], [slot]
```

Speichert das aktuell bearbeitete Transkript.

Todo Ersetzt aktuell die zugehörige JSON-Datei. Dies soll in Zukunft einen 'UPDATE'-Befehl an die **Datenbank** senden, um den bestehenden Eintrag zu aktualisieren.

7.6.3.25 saveTranscriptionToJsonAs

```
void MainWindow::saveTranscriptionToJsonAs ( )  [private], [slot]
```

Öffnet einen Dateidialog, um das Transkript unter einem neuen Namen zu speichern.

Todo Statt als neue Datei zu speichern, soll dies zukünftig einen 'INSERT'-Befehl an die **Datenbank** senden, um einen neuen Eintrag zu erzeugen.

7.6.3.26 setMeetingName

```
void MainWindow::setMeetingName (
            const QString & name )  [private], [slot]
```

Setzt den Namen für das aktuelle Meeting.

7.6.3.27 setStatus

```
void MainWindow::setStatus (
            const QString & text,
            bool keep = false )  [private], [slot]
```

Zeigt eine Nachricht in der Statusleiste an.

7.6.3.28 setupUI()

```
void MainWindow::setupUI ( )  [private]
```

Erstellt und arrangiert alle UI-Widgets.

7.6.3.29 updateUiForCurrentMeeting()

```
void MainWindow::updateUiForCurrentMeeting ( )  [private]
```

Aktualisiert den Zustand der UI (Buttons, Labels) basierend auf dem aktuellen Meeting-Status.

7.6.3.30 updateUndoRedoState

```
void MainWindow::updateUndoRedoState ( )  [private], [slot]
```

Aktualisiert den Zustand der Undo/Redo-Buttons.

7.6.4 Member Data Documentation

7.6.4.1 assignNamesButton

```
QPushButton* MainWindow::assignNamesButton  [private]
```

7.6.4.2 buttonLayout

```
QHBoxLayout* MainWindow::buttonLayout  [private]
```

7.6.4.3 editTextButton

```
QPushButton* MainWindow::editTextButton  [private]
```

7.6.4.4 elapsedTime

```
QElapsedTimer MainWindow::elapsedTime  [private]
```

7.6.4.5 generateTagsButton

```
QPushButton* MainWindow::generateTagsButton  [private]
```

7.6.4.6 leftPanel

```
QWidget* MainWindow::leftPanel  [private]
```

7.6.4.7 m_actionClose

```
QAction* MainWindow::m_actionClose  [private]
```

7.6.4.8 m_actionOpen

```
QAction* MainWindow::m_actionOpen  [private]
```

7.6.4.9 m_actionRestoreOriginal

```
QAction* MainWindow::m_actionRestoreOriginal  [private]
```

7.6.4.10 m_actionSave

```
QAction* MainWindow::m_actionSave  [private]
```

7.6.4.11 m_actionSaveAs

```
QAction* MainWindow::m_actionSaveAs  [private]
```

7.6.4.12 m_actionSetMeetingName

```
QAction* MainWindow::m_actionSetMeetingName  [private]
```

7.6.4.13 m_asrManager

```
AsrProcessManager* MainWindow::m_asrManager  [private]
```

Manager für den ASR-Python-Prozess.

7.6.4.14 m_captureThread

```
CaptureThread* MainWindow::m_captureThread  [private]
```

Thread für die plattformspezifische Audio-Aufnahme.

7.6.4.15 m_currentAudioPath

```
QString MainWindow::m_currentAudioPath  [private]
```

Pfad zur zuletzt gespeicherten Audiodatei.

7.6.4.16 m_currentMeetingDateTime

```
QString MainWindow::m_currentMeetingDateTime  [private]
```

Zeitstempel des aktuellen Meetings.

7.6.4.17 m_currentMeetingName

```
QString MainWindow::m_currentMeetingName  [private]
```

Name des aktuellen Meetings (wird bei Aufnahme/Laden gesetzt).

7.6.4.18 m_fileManager

```
FileManager* MainWindow::m_fileManager  [private]
```

Manager für alle Dateizugriffe.

7.6.4.19 m_redoAction

```
QAction* MainWindow::m_redoAction  [private]
```

Menü-Aktion für Redo.

7.6.4.20 m_redoStack

```
QStack<QJsonDocument> MainWindow::m_redoStack  [private]
```

Stapel für die Redo-Zustände.

7.6.4.21 m_reinstallPythonAction

```
QAction* MainWindow::m_reinstallPythonAction  [private]
```

7.6.4.22 m_script

```
Transcription* MainWindow::m_script  [private]
```

Das zentrale Datenmodell für das Transkript.

7.6.4.23 m_settingsAction

```
QAction* MainWindow::m_settingsAction  [private]
```

7.6.4.24 m_speakerEditorDialog

```
SpeakerEditorDialog* MainWindow::m_speakerEditorDialog  [private]
```

7.6.4.25 m_tagGenerator

```
TagGeneratorManager* MainWindow::m_tagGenerator  [private]
```

Manager für den Tag-Generator-Python-Prozess.

7.6.4.26 m_textEditorDialog

```
TextEditorDialog* MainWindow::m_textEditorDialog  [private]
```

7.6.4.27 m_undoAction

```
QAction* MainWindow::m_undoAction  [private]
```

Menü-Aktion für Undo.

7.6.4.28 m_undoStack

```
QStack<QJsonDocument> MainWindow::m_undoStack  [private]
```

Stapel für die Undo-Zustände.

7.6.4.29 m_wavWriter

```
WavWriterThread* MainWindow::m_wavWriter  [private]
```

Thread zum Schreiben der WAV-Dateien.

7.6.4.30 mainLayout

```
QVBoxLayout* MainWindow::mainLayout  [private]
```

7.6.4.31 meetingList

```
QListWidget* MainWindow::meetingList  [private]
```

7.6.4.32 nameLabel

```
QLabel* MainWindow::nameLabel  [private]
```

7.6.4.33 pluginProcess

```
QProcess* MainWindow::pluginProcess  [private]
```

Platzhalter für einen möglichen IPC-Prozess.

7.6.4.34 pollTimer

```
QTimer* MainWindow::pollTimer  [private]
```

7.6.4.35 rightPanel

```
QWidget* MainWindow::rightPanel  [private]
```

7.6.4.36 saveAudioButton

```
QPushButton* MainWindow::saveAudioButton  [private]
```

7.6.4.37 savePDFButton

```
QPushButton* MainWindow::savePDFButton  [private]
```

7.6.4.38 searchBox

```
QLineEdit* MainWindow::searchBox  [private]
```

7.6.4.39 splitter

```
QSplitter* MainWindow::splitter  [private]
```

7.6.4.40 startButton

```
QPushButton* MainWindow::startButton  [private]
```

7.6.4.41 statusLabel

```
QLabel* MainWindow::statusLabel  [private]
```

7.6.4.42 statusTimer

```
QTimer* MainWindow::statusTimer  [private]
```

7.6.4.43 stopButton

```
QPushButton* MainWindow::stopButton  [private]
```

7.6.4.44 timeLabel

```
QLabel* MainWindow::timeLabel  [private]
```

7.6.4.45 timeUpdateTimer

```
QTimer* MainWindow::timeUpdateTimer  [private]
```

7.6.4.46 transcriptView

```
QTextEdit* MainWindow::transcriptView  [private]
```

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

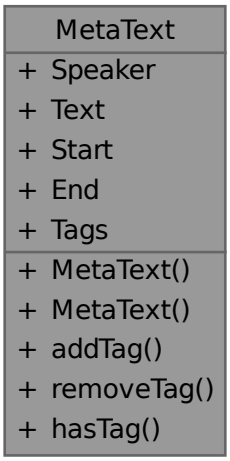
- mainwindow.h
- mainwindow.cpp

7.7 MetaText Struct Reference

Eine einfache Datenstruktur, die ein einzelnes Segment eines Transkripts repräsentiert.

`#include <transcription.h>`

Collaboration diagram for MetaText:



Public Member Functions

- MetaText ()=default
- MetaText (const QString &start, const QString &end, const QString &speaker, const QString &text)
- void addTag (const QString &tag)
- void removeTag (const QString &tag)
- bool hasTag (const QString &tag) const

Public Attributes

- QString Speaker

  Der Name des Sprechers für dieses Segment.
- QString Text

  Der transkribierte Text des Segments.
- QString Start

  Start-Zeitstempel des Segments (als String in Sekunden).
- QString End

  End-Zeitstempel des Segments (als String in Sekunden).
- QStringList Tags

  Eine Liste von Tags, die diesem spezifischen Segment zugeordnet sind.

7.7.1 Detailed Description

Eine einfache Datenstruktur, die ein einzelnes Segment eines Transkripts repräsentiert.

Enthält den Text selbst sowie Metadaten wie Sprecher, Zeitstempel und zugeordnete Tags.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 MetaText() [1/2]

```
MetaText::MetaText ( )  [default]
```

7.7.2.2 MetaText() [2/2]

```
MetaText::MetaText (
            const QString & start,
            const QString & end,
            const QString & speaker,
            const QString & text )  [inline]
```

7.7.3 Member Function Documentation

7.7.3.1 addTag()

```
void MetaText::addTag (
            const QString & tag )  [inline]
```

7.7.3.2 hasTag()

```
bool MetaText::hasTag (
            const QString & tag ) const  [inline]
```

7.7.3.3 removeTag()

```
void MetaText::removeTag (
            const QString & tag )  [inline]
```

7.7.4 Member Data Documentation

7.7.4.1 End

```
QString MetaText::End
```

End-Zeitstempel des Segments (als String in Sekunden).

7.7.4.2 Speaker

```
QString MetaText::Speaker
```

Der Name des Sprechers für dieses Segment.

7.7.4.3 Start

```
QString MetaText::Start
```

Start-Zeitstempel des Segments (als String in Sekunden).

7.7.4.4 Tags

```
QStringList MetaText::Tags
```

Eine Liste von Tags, die diesem spezifischen Segment zugeordnet sind.

7.7.4.5 Text

```
QString MetaText::Text
```

Der transkribierte Text des Segments.

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

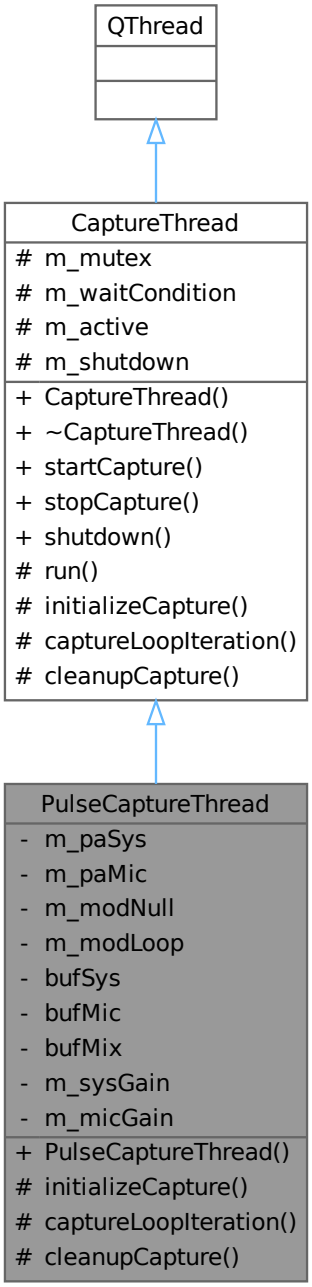
- transcription.h

7.8 PulseCaptureThread Class Reference

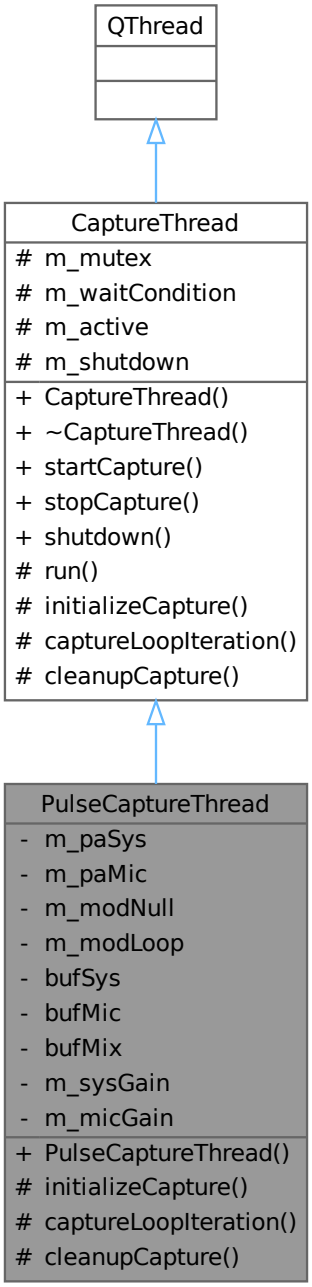
Eine konkrete Implementierung von CaptureThread für Linux-Systeme mit PulseAudio.

```
#include <pulsecapturethread.h>
```

Inheritance diagram for PulseCaptureThread:



Collaboration diagram for PulseCaptureThread:



Public Member Functions

- PulseCaptureThread (QObject ∗parent=nullptr)

  Standard-Konstruktor.

Public Member Functions inherited from CaptureThread

- CaptureThread (QObject ∗parent=nullptr)

Standard-Konstruktor.

- virtual ~CaptureThread ()=default

  Virtueller Destruktor.

- void startCapture ()

  Startet eine neue Aufnahme-Session. Diese Methode ist thread-sicher und weckt den run()-Loop auf, um mit der Aufnahme zu beginnen.

- virtual void stopCapture ()

  Fordert das Beenden der aktuellen Aufnahme-Session an. Dies ist ein nicht-blockierender Aufruf. Der Thread beendet die Aufnahmeschleife so bald wie möglich.

- void shutdown ()

  Beendet den Thread vollständig und wartet auf dessen Terminierung. Dies ist ein blockierender Aufruf, der sicherstellt, dass alle Ressourcen freigegeben werden.

Protected Member Functions

- bool initializeCapture () override

  Initialisiert die PulseAudio-Aufnahmeumgebung.

- void captureLoopIteration () override

  Führt eine einzelne Iteration der Aufnahmeschleife aus.

- void cleanupCapture () override

  Gibt alle für die Aufnahme erstellten PulseAudio-Ressourcen frei.

Protected Member Functions inherited from CaptureThread

- void run () override

  Die Hauptfunktion des Threads, die von QThread aufgerufen wird. Sie implementiert die Zustandsmaschine für den Aufnahme-Lebenszyklus.

Private Attributes

- pa_simple * m_paSys

  Handle für den PulseAudio-Stream der System-Sounds.

- pa_simple * m_paMic

  Handle für den PulseAudio-Stream des Mikrofons.

- int m_modNull

  ID des geladenen module-null-sink.

- int m_modLoop

  ID des geladenen module-loopback.

- std::vector< float > bufSys
- std::vector< float > bufMic
- std::vector< float > bufMix

  Puffer für die Audio-Samples.

- float m_sysGain
- float m_micGain

  Verstärkungsfaktoren für System- und Mikrofon-Audio.

Additional Inherited Members

Signals inherited from CaptureThread

- void pcmChunkReady (QList< float > chunk)

  Wird gesendet, wenn ein neuer Block von Audio-Daten (PCM-Samples) bereitsteht.
- void started ()

  Wird gesendet, unmittelbar nachdem die plattformspezifische Initialisierung erfolgreich war und die Aufnahmeschleife beginnt.
- void stopped ()

  Wird gesendet, nachdem die Aufnahmeschleife beendet und die Aufräumarbeiten abgeschlossen sind.

Protected Attributes inherited from CaptureThread

- QMutex m_mutex

  Schützt den Zugriff auf den Zustand des Threads.
- QWaitCondition m_waitCondition

  Lässt den Thread schlafen, wenn er inaktiv ist.
- std::atomic< bool > m_active

  Steuert die innere Aufnahmeschleife (start/stop).
- std::atomic< bool > m_shutdown

  Signalisiert dem Thread, sich komplett zu beenden.

7.8.1 Detailed Description

Eine konkrete Implementierung von CaptureThread für Linux-Systeme mit PulseAudio.

Diese Klasse implementiert die Audio-Aufnahme durch die Interaktion mit dem PulseAudio-Soundserver. Sie verwendet Kommandozeilen-Aufrufe von `pactl`, um eine virtuelle Audio-Senke (null-sink) und ein Loopback-Gerät für das Mikrofon zu erstellen. Dies ermöglicht das getrennte Abgreifen von System-Audio und Mikrofon-Audio. Die eigentliche Aufnahme der Audiodaten erfolgt über die "Simple API" von PulseAudio (libpulse-simple).

7.8.2 Constructor & Destructor Documentation

7.8.2.1 PulseCaptureThread()

```
PulseCaptureThread::PulseCaptureThread (
            QObject * parent = nullptr )  [explicit]
```

Standard-Konstruktor.

Parameters

parent	Das QObject-Elternteil für die Speicherverwaltung.

7.8.3 Member Function Documentation

7.8.3.1 captureLoopIteration()

```
void PulseCaptureThread::captureLoopIteration ( )  [override], [protected], [virtual]
```

Führt eine einzelne Iteration der Aufnahmeschleife aus.

Liest Audio-Daten vom System- und Mikrofon-Stream, mischt diese unter Berücksichtigung der Gain-Faktoren und sendet das Ergebnis über das pcmChunkReady-Signal.

Implements CaptureThread.

7.8.3.2 cleanupCapture()

```
void PulseCaptureThread::cleanupCapture ( )  [override], [protected], [virtual]
```

Gibt alle für die Aufnahme erstellten PulseAudio-Ressourcen frei.

Schließt die `pa_simple` Streams und entlädt die zuvor per `pactl` geladenen Kernel-Module.

Implements CaptureThread.

7.8.3.3 initializeCapture()

```
bool PulseCaptureThread::initializeCapture ( )  [override], [protected], [virtual]
```

Initialisiert die PulseAudio-Aufnahmeumgebung.

Findet die Standard-Geräte, lädt die `module-null-sink` und `module-loopback` Kernel-Module via `pactl` und öffnet zwei `pa_simple` Streams zum Mitschneiden.

Returns

true bei Erfolg, andernfalls false.

Implements CaptureThread.

7.8.4 Member Data Documentation

7.8.4.1 bufMic

```
std::vector<float> PulseCaptureThread::bufMic  [private]
```

7.8.4.2 bufMix

```
std::vector<float> PulseCaptureThread::bufMix  [private]
```

Puffer für die Audio-Samples.

7.8.4.3 bufSys

```
std::vector<float> PulseCaptureThread::bufSys  [private]
```

7.8.4.4 m_micGain

```
float PulseCaptureThread::m_micGain  [private]
```

Verstärkungsfaktoren für System- und Mikrofon-Audio.

7.8.4.5 m_modLoop

```
int PulseCaptureThread::m_modLoop  [private]
```

ID des geladenen module-loopback.

7.8.4.6 m_modNull

```
int PulseCaptureThread::m_modNull  [private]
```

ID des geladenen module-null-sink.

7.8.4.7 m_paMic

```
pa_simple* PulseCaptureThread::m_paMic  [private]
```

Handle für den PulseAudio-Stream des Mikrofons.

7.8.4.8 m_paSys

```
pa_simple* PulseCaptureThread::m_paSys  [private]
```

Handle für den PulseAudio-Stream der System-Sounds.

7.8.4.9 m_sysGain

```
float PulseCaptureThread::m_sysGain  [private]
```

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

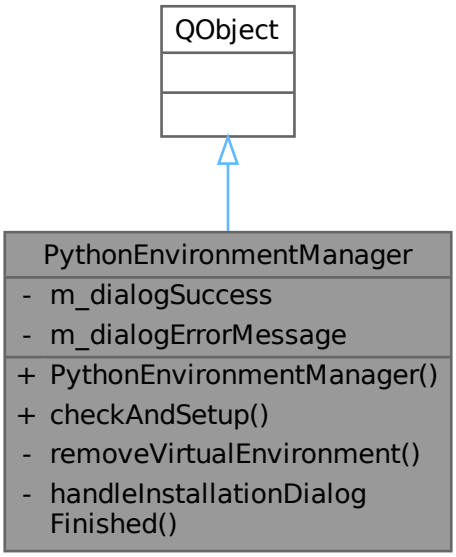
- pulsecapturethread.h
- pulsecapturethread.cpp

7.9 PythonEnvironmentManager Class Reference

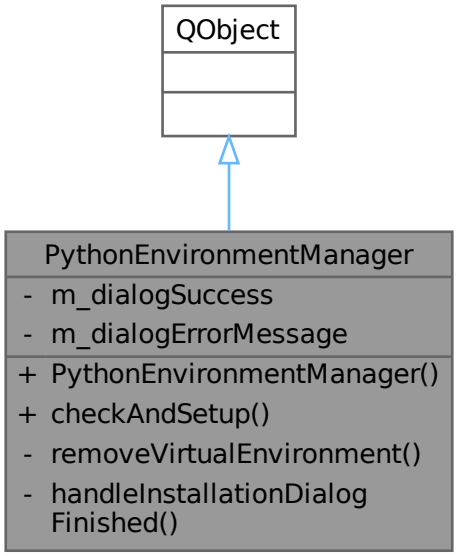
Verwaltet die Python-Umgebung und deren Installation/Prüfung.

`#include <pythonenvironmentmanager.h>`

Inheritance diagram for PythonEnvironmentManager:



Collaboration diagram for PythonEnvironmentManager:



Public Member Functions

- PythonEnvironmentManager (QObject *parent=nullptr)
- bool checkAndSetup (bool forceReinstall=false, QWidget *parentWidget=nullptr)

  Überprüft die Python-Umgebung und führt bei Bedarf eine Installation durch.

Private Slots

- void handleInstallationDialogFinished (bool success, const QString &errorMessage)

  Slot zur Verarbeitung des `installationFinished`-Signals vom `InstallationDialog`.

Private Member Functions

- bool removeVirtualEnvironment (const QString &venvPath)

  Löscht rekursiv eine vorhandene virtuelle Python-Umgebung.

Private Attributes

- bool m_dialogSuccess

  Speichert das Erfolgsergebnis des `InstallationDialog`.
- QString m_dialogErrorMessage

  Speichert die Fehlermeldung des `InstallationDialog`.

7.9.1 Detailed Description

Verwaltet die Python-Umgebung und deren Installation/Prüfung.

Diese Klasse kapselt die gesamte Logik, die für die Überprüfung, Einrichtung, und optionale Neuinstallation der Python-Umgebung (insbesondere einer virtuellen Umgebung und ihrer Abhängigkeiten) erforderlich ist. Sie interagiert mit dem InstallationDialog, um den Fortschritt anzuzeigen und Rückmeldungen zu geben.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 PythonEnvironmentManager()

```
PythonEnvironmentManager::PythonEnvironmentManager (
            QObject * parent = nullptr )  [explicit]
```

7.9.3 Member Function Documentation

7.9.3.1 checkAndSetup()

```
bool PythonEnvironmentManager::checkAndSetup (
            bool forceReinstall = false,
            QWidget * parentWidget = nullptr )
```

Überprüft die Python-Umgebung und führt bei Bedarf eine Installation durch.

Diese Methode ist der zentrale Einstiegspunkt. Sie prüft, ob ein gültiger Python-Pfad vorhanden ist. Falls nicht oder wenn `forceReinstall` true ist, wird eine Neuinstallation initiiert. Ein modaler `InstallationDialog` wird angezeigt, um den Fortschritt darzustellen.

Parameters

forceReinstall	Wenn `true`, wird die virtuelle Python-Umgebung gelöscht und neu aufgebaut, auch wenn bereits ein Pfad konfiguriert ist.
parentWidget	Ein optionales Eltern-Widget für den `InstallationDialog`. Dies ist wichtig für die korrekte Positionierung und Modalität des Dialogs.

Returns

`true`, wenn die Umgebung erfolgreich konfiguriert wurde, `false` bei Fehler oder Abbruch.

7.9.3.2 handleInstallationDialogFinished

```
void PythonEnvironmentManager::handleInstallationDialogFinished (
            bool success,
            const QString & errorMessage )  [private], [slot]
```

Slot zur Verarbeitung des `installationFinished`-Signals vom `InstallationDialog`.

Dieser Slot wird aufgerufen, wenn der `InstallationDialog` seine Arbeit beendet hat. Er puffert das Ergebnis intern, um es nach dem Schließen des Dialogs auszuwerten.

Parameters

success	`true`, wenn der Prozess im Dialog erfolgreich war.
errorMessage	Eine eventuelle Fehlermeldung vom Dialog.

7.9.3.3 removeVirtualEnvironment()

```
bool PythonEnvironmentManager::removeVirtualEnvironment (
            const QString & venvPath )  [private]
```

Löscht rekursiv eine vorhandene virtuelle Python-Umgebung.

Parameters

venvPath	Der absolute Pfad zum Verzeichnis der virtuellen Umgebung.

Returns

`true` bei erfolgreichem Löschen oder wenn das Verzeichnis nicht existiert.

7.9.4 Member Data Documentation

7.9.4.1 m_dialogErrorMessage

```
QString PythonEnvironmentManager::m_dialogErrorMessage  [private]
```

Speichert die Fehlermeldung des `InstallationDialog`.

7.9.4.2 m_dialogSuccess

```
bool PythonEnvironmentManager::m_dialogSuccess  [private]
```

Speichert das Erfolgsergebnis des `InstallationDialog`.

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

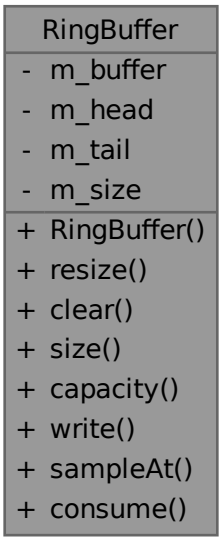
- pythonenvironmentmanager.h
- pythonenvironmentmanager.cpp

7.10 RingBuffer Class Reference

Eine einfache und effiziente Ringpuffer-Implementierung für float-Werte.

`#include <ringbuffer.h>`

Collaboration diagram for RingBuffer:



Public Member Functions

- RingBuffer (size_t capacity=0)

  Erstellt einen Puffer mit einer festen maximalen Kapazität.
- void resize (size_t capacity)

  Ändert die Größe des Puffers und löscht seinen Inhalt.
- void clear ()

  Löscht den Inhalt des Puffers, indem die Zeiger zurückgesetzt werden.
- size_t size () const

  Gibt die Anzahl der aktuell im Puffer befindlichen Elemente zurück.
- size_t capacity () const

  Gibt die maximale Kapazität des Puffers zurück.
- void write (const float *data, size_t count)

  Schreibt neue Daten in den Puffer.
- float sampleAt (double pos) const

  Liest einen Sample-Wert an einer bestimmten Fließkomma-Position.
- void consume (size_t count)

  "Konsumiert" eine Anzahl von Samples, indem der Lesezeiger verschoben wird.

Private Attributes

- std::vector< float > m_buffer

  Der zugrundeliegende Speicher für die Pufferdaten.
- size_t m_head = 0

  Die Position (Index) für den nächsten Schreibvorgang.
- size_t m_tail = 0

  Die Position (Index) des ältesten, noch gültigen Samples.
- size_t m_size = 0

  Die aktuelle Anzahl der gültigen Samples im Puffer.

7.10.1 Detailed Description

Eine einfache und effiziente Ringpuffer-Implementierung für float-Werte.

Diese Klasse implementiert einen "First-In, First-Out" (FIFO) Puffer mit einer festen Kapazität. Wenn der Puffer voll ist und neue Daten geschrieben werden, werden die ältesten Daten überschrieben. Dies ist nützlich für das Streamen von Audiodaten, wo nur die letzten paar Sekunden an Daten relevant sind. Da dies eine Header-only-↩ Klasse ist, sind alle Methoden inline implementiert.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 RingBuffer()

```
RingBuffer::RingBuffer (
            size_t capacity = 0 )  [inline], [explicit]
```

Erstellt einen Puffer mit einer festen maximalen Kapazität.

Parameters

capacity	Die maximale Anzahl an float-Werten, die der Puffer halten kann.

7.10.3 Member Function Documentation

7.10.3.1 capacity()

```
size_t RingBuffer::capacity ( ) const  [inline]
```

Gibt die maximale Kapazität des Puffers zurück.

Returns

Die maximale Anzahl an Elementen, die der Puffer halten kann.

7.10.3.2 clear()

```
void RingBuffer::clear ( )  [inline]
```

Löscht den Inhalt des Puffers, indem die Zeiger zurückgesetzt werden.

7.10.3.3 consume()

```
void RingBuffer::consume (
            size_t count )  [inline]
```

"Konsumiert" eine Anzahl von Samples, indem der Lesezeiger verschoben wird.

Entfernt effektiv die ältesten `count` Elemente aus dem Puffer.

Parameters

count	Die Anzahl der zu entfernenden Elemente.

7.10.3.4 resize()

```
void RingBuffer::resize (
            size_t capacity )  [inline]
```

Ändert die Größe des Puffers und löscht seinen Inhalt.

Parameters

capacity	Die neue maximale Kapazität.

7.10.3.5 sampleAt()

```
float RingBuffer::sampleAt (
            double pos ) const  [inline]
```

Liest einen Sample-Wert an einer bestimmten Fließkomma-Position.

Diese Funktion verwendet lineare Interpolation zwischen zwei Samples, um einen Wert an einer nicht-ganzzahligen Position zu schätzen. Dies ist sehr nützlich für Audio-Resampling.

Parameters

pos	Die Fließkomma-Position des gewünschten Samples relativ zum Pufferanfang.

Returns

Der interpolierte Sample-Wert.

7.10.3.6 size()

```
size_t RingBuffer::size ( ) const  [inline]
```

Gibt die Anzahl der aktuell im Puffer befindlichen Elemente zurück.

Returns

Die aktuelle Füllmenge.

7.10.3.7 write()

```
void RingBuffer::write (
            const float * data,
            size_t count )  [inline]
```

Schreibt neue Daten in den Puffer.

Fügt die Daten am Schreibzeiger hinzu. Wenn der Puffer voll ist, werden die ältesten Daten überschrieben.

Parameters

data	Ein Zeiger auf das Array mit den zu schreibenden Daten.
count	Die Anzahl der zu schreibenden float-Werte.

7.10.4 Member Data Documentation

7.10.4.1 m_buffer

```
std::vector<float> RingBuffer::m_buffer  [private]
```

Der zugrundeliegende Speicher für die Pufferdaten.

7.10.4.2 m_head

```
size_t RingBuffer::m_head = 0  [private]
```

Die Position (Index) für den nächsten Schreibvorgang.

7.10.4.3 m_size

```
size_t RingBuffer::m_size = 0  [private]
```

Die aktuelle Anzahl der gültigen Samples im Puffer.

7.10.4.4 m_tail

```
size_t RingBuffer::m_tail = 0  [private]
```

Die Position (Index) des ältesten, noch gültigen Samples.

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

- ringbuffer.h

7.11 SettingsWizard Class Reference

Ein Dialogfenster zur Bearbeitung der Anwendungseinstellungen.

```
#include <settingswizard.h>
```

Inheritance diagram for SettingsWizard:

Collaboration diagram for SettingsWizard:

Public Member Functions

- SettingsWizard (QWidget ∗parent=nullptr)

  Konstruktor, der die UI des Dialogs aufbaut und mit Werten füllt.

Private Slots

- void saveSettings ()

  Speichert die aktuellen Werte aller UI-Elemente in die QSettings.
- void updateDurationLabel (int value)

  Aktualisiert ein Label, das die Puffergröße in Sekunden anzeigt.
- void syncSysGainSlider (double value)

Synchronisiert den System-Gain-Slider mit der SpinBox (logarithmische Skala).

- void syncMicGainSlider (double value)

  Synchronisiert den Mikrofon-Gain-Slider mit der SpinBox (logarithmische Skala).
- void syncSysGainSpin (int sliderValue)

  Synchronisiert die System-Gain-SpinBox mit dem Slider.
- void syncMicGainSpin (int sliderValue)

  Synchronisiert die Mikrofon-Gain-SpinBox mit dem Slider.

Private Member Functions

- int validateBufferSize (int kb)

  Stellt sicher, dass die Puffergröße innerhalb eines gültigen Bereichs liegt.

Private Attributes

- QLineEdit * pythonEdit

  Eingabefeld für den Python-Pfad.
- QLineEdit * scriptEdit

  Eingabefeld für den ASR-Skript-Pfad.
- QLineEdit * wavEdit

  Eingabefeld für den Wav-Datei-Pfad.
- QLineEdit * asrWavEdit

  Eingabefeld für den ASR-Wav-Datei-Pfad.
- QLineEdit * meetingEdit

  Eingabefeld für den Meetings-Pfad.
- QSlider * bufferSlider

  Slider zur Einstellung der Audio-Puffergröße.
- QLabel * durationLabel

  Label zur Anzeige der Pufferdauer in Sekunden.
- QDoubleSpinBox * sysGainSpin

  SpinBox für den System-Audio-Verstärkungsfaktor.
- QSlider * sysGainSlider

  Slider für den System-Audio-Verstärkungsfaktor.
- QDoubleSpinBox * micGainSpin

  SpinBox für den Mikrofon-Verstärkungsfaktor.
- QSlider * micGainSlider

  Slider für den Mikrofon-Verstärkungsfaktor.
- QSpinBox * pdfHeadlineSpin

  SpinBox für die Schriftgröße der PDF-Überschrift.
- QSpinBox * pdfBodySpin

  SpinBox für die Schriftgröße des PDF-Haupttextes.
- QSpinBox * pdfMetaSpin

  SpinBox für die Schriftgröße der PDF-Metadaten.
- QSpinBox * marginTopSpin

  SpinBox für den oberen Seitenrand des PDFs.
- QSpinBox * marginRightSpin

  SpinBox für den rechten Seitenrand des PDFs.
- QSpinBox * marginBottomSpin

  SpinBox für den unteren Seitenrand des PDFs.
- QSpinBox * marginLeftSpin

  SpinBox für den linken Seitenrand des PDFs.
- QFontComboBox * fontFamilyCombo

  Auswahlbox für die PDF-Schriftfamilie.

7.11.1 Detailed Description

Ein Dialogfenster zur Bearbeitung der Anwendungseinstellungen.

Dieser Dialog bietet eine grafische Oberfläche, um diverse Parameter der Anwendung zu konfigurieren. Die Werte werden aus einem QSettings-Objekt geladen und beim Speichern wieder dorthin zurückgeschrieben, um sie persistent zu machen.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 SettingsWizard()

```
SettingsWizard::SettingsWizard (
            QWidget * parent = nullptr )  [explicit]
```

Konstruktor, der die UI des Dialogs aufbaut und mit Werten füllt.

Parameters

parent	Das QObject-Elternteil für die Speicherverwaltung.

7.11.3 Member Function Documentation

7.11.3.1 saveSettings

```
void SettingsWizard::saveSettings ( )  [private], [slot]
```

Speichert die aktuellen Werte aller UI-Elemente in die QSettings.

7.11.3.2 syncMicGainSlider

```
void SettingsWizard::syncMicGainSlider (
            double value )  [private], [slot]
```

Synchronisiert den Mikrofon-Gain-Slider mit der SpinBox (logarithmische Skala).

7.11.3.3 syncMicGainSpin

```
void SettingsWizard::syncMicGainSpin (
            int sliderValue )  [private], [slot]
```

Synchronisiert die Mikrofon-Gain-SpinBox mit dem Slider.

7.11.3.4 syncSysGainSlider

```
void SettingsWizard::syncSysGainSlider (
            double value )  [private], [slot]
```

Synchronisiert den System-Gain-Slider mit der SpinBox (logarithmische Skala).

7.11.3.5 syncSysGainSpin

```
void SettingsWizard::syncSysGainSpin (
            int sliderValue )  [private], [slot]
```

Synchronisiert die System-Gain-SpinBox mit dem Slider.

7.11.3.6 updateDurationLabel

```
void SettingsWizard::updateDurationLabel (
            int value )  [private], [slot]
```

Aktualisiert ein Label, das die Puffergröße in Sekunden anzeigt.

7.11.3.7 validateBufferSize()

```
int SettingsWizard::validateBufferSize (
            int kb )  [private]
```

Stellt sicher, dass die Puffergröße innerhalb eines gültigen Bereichs liegt.

Parameters

kb	Die zu validierende Größe in Kilobyte.

Returns

Die validierte und ggf. korrigierte Größe.

7.11.4 Member Data Documentation

7.11.4.1 asrWavEdit

```
QLineEdit* SettingsWizard::asrWavEdit  [private]
```

Eingabefeld für den ASR-Wav-Datei-Pfad.

7.11.4.2 bufferSlider

```
QSlider* SettingsWizard::bufferSlider  [private]
```

Slider zur Einstellung der Audio-Puffergröße.

7.11.4.3 durationLabel

```
QLabel* SettingsWizard::durationLabel  [private]
```

Label zur Anzeige der Pufferdauer in Sekunden.

7.11.4.4 fontFamilyCombo

```
QFontComboBox* SettingsWizard::fontFamilyCombo  [private]
```

Auswahlbox für die PDF-Schriftfamilie.

7.11.4.5 marginBottomSpin

```
QSpinBox* SettingsWizard::marginBottomSpin  [private]
```

SpinBox für den unteren Seitenrand des PDFs.

7.11.4.6 marginLeftSpin

```
QSpinBox* SettingsWizard::marginLeftSpin  [private]
```

SpinBox für den linken Seitenrand des PDFs.

7.11.4.7 marginRightSpin

```
QSpinBox* SettingsWizard::marginRightSpin  [private]
```

SpinBox für den rechten Seitenrand des PDFs.

7.11.4.8 marginTopSpin

```
QSpinBox* SettingsWizard::marginTopSpin  [private]
```

SpinBox für den oberen Seitenrand des PDFs.

7.11.4.9 meetingEdit

```
QLineEdit* SettingsWizard::meetingEdit  [private]
```

Eingabefeld für den Meetings-Pfad.

7.11.4.10 micGainSlider

```
QSlider* SettingsWizard::micGainSlider  [private]
```

Slider für den Mikrofon-Verstärkungsfaktor.

7.11.4.11 micGainSpin

```
QDoubleSpinBox* SettingsWizard::micGainSpin  [private]
```

SpinBox für den Mikrofon-Verstärkungsfaktor.

7.11.4.12 pdfBodySpin

```
QSpinBox* SettingsWizard::pdfBodySpin  [private]
```

SpinBox für die Schriftgröße des PDF-Haupttextes.

7.11.4.13 pdfHeadlineSpin

```
QSpinBox* SettingsWizard::pdfHeadlineSpin  [private]
```

SpinBox für die Schriftgröße der PDF-Überschrift.

7.11.4.14 pdfMetaSpin

```
QSpinBox* SettingsWizard::pdfMetaSpin  [private]
```

SpinBox für die Schriftgröße der PDF-Metadaten.

7.11.4.15 pythonEdit

```
QLineEdit* SettingsWizard::pythonEdit  [private]
```

Eingabefeld für den Python-Pfad.

7.11.4.16 scriptEdit

```
QLineEdit* SettingsWizard::scriptEdit  [private]
```

Eingabefeld für den ASR-Skript-Pfad.

7.11.4.17 sysGainSlider

```
QSlider* SettingsWizard::sysGainSlider  [private]
```

Slider für den System-Audio-Verstärkungsfaktor.

7.11.4.18 sysGainSpin

```
QDoubleSpinBox* SettingsWizard::sysGainSpin  [private]
```

SpinBox für den System-Audio-Verstärkungsfaktor.

7.11.4.19 wavEdit

```
QLineEdit* SettingsWizard::wavEdit  [private]
```

Eingabefeld für den Wav-Datei-Pfad.

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

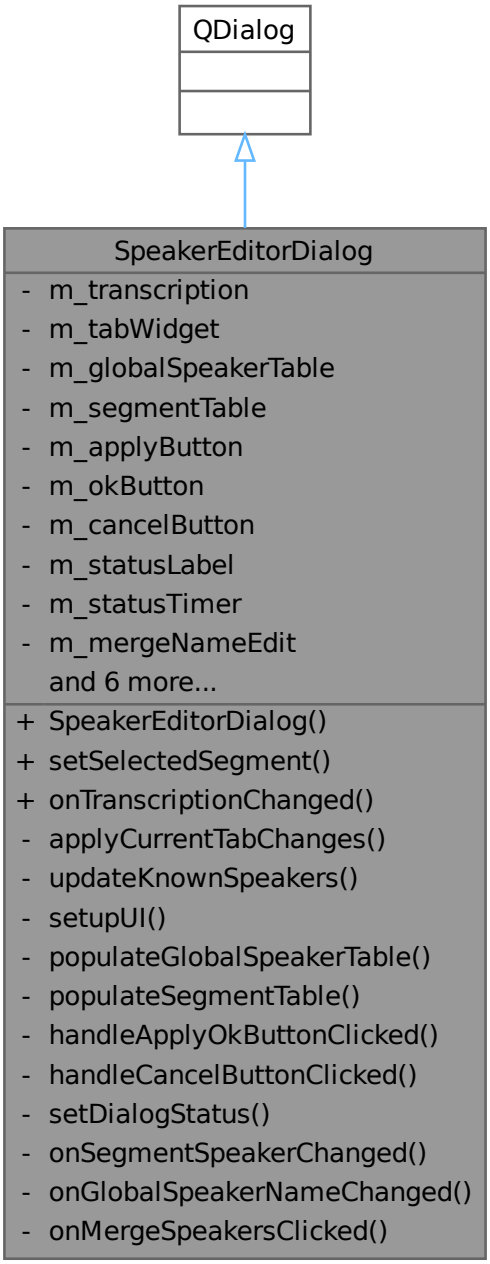
- settingswizard.h
- settingswizard.cpp

7.12 SpeakerEditorDialog Class Reference

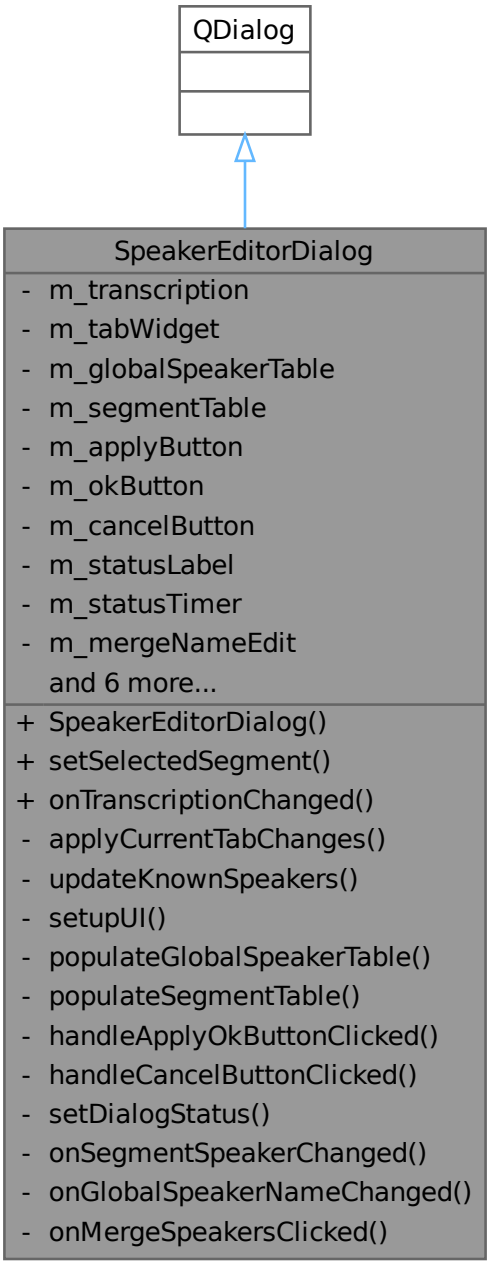
Ein nicht-modaler Dialog zur Bearbeitung von Sprecherinformationen im Transkript.

```
#include <speakereditordialog.h>
```

Inheritance diagram for SpeakerEditorDialog:



Collaboration diagram for SpeakerEditorDialog:



Public Slots

- void onTranscriptionChanged ()

  Aktualisiert die Ansichten des Dialogs, wenn sich das zugrundeliegende Transcription-Objekt ändert.

Public Member Functions

- SpeakerEditorDialog (Transcription *transcription, QWidget *parent=nullptr)

Konstruktor, der den Dialog initialisiert und mit dem Datenmodell verknüpft.

- void setSelectedSegment (const QString &start, const QString &end)

  Ermöglicht das programmatische Vor-Auswählen eines Segments im Dialog.

Private Slots

- void setupUI ()

  Erstellt die Benutzeroberfläche und verbindet die internen Signale und Slots.
- void populateGlobalSpeakerTable ()

  Füllt die Tabelle für die globalen Sprechernamen mit Daten.
- void populateSegmentTable ()

  Füllt die Tabelle für die einzelnen Textsegmente mit Daten.
- void handleApplyOkButtonClicked ()

  Slot für die "Anwenden" und "OK" Buttons. Übernimmt die Änderungen.
- void handleCancelButtonClicked ()

  Slot für den "Abbrechen" Button. Schließt den Dialog ohne Änderungen.
- void setDialogStatus (const QString &text, bool temporary=true)

  Zeigt eine temporäre oder permanente Nachricht in der Statuszeile des Dialogs an.
- void onSegmentSpeakerChanged (int index)

  Reagiert auf eine Änderung in der Sprecher-ComboBox eines Segments.
- void onGlobalSpeakerNameChanged (const QString &text)

  Reagiert auf eine Textänderung im Eingabefeld für einen globalen Sprechernamen.
- void onMergeSpeakersClicked ()

  Führt die ausgewählten Sprecher unter einem neuen Namen zusammen.

Private Member Functions

- void applyCurrentTabChanges ()

  Wendet die im aktuell aktiven Tab vorgenommenen Änderungen auf das Transcription-Objekt an.
- void updateKnownSpeakers ()

  Aktualisiert die interne Liste aller bekannten Sprechernamen aus dem Transkript.

Private Attributes

- QPointer< Transcription > m_transcription

  Ein sicherer Zeiger auf das Transkript-Datenmodell.
- QTabWidget * m_tabWidget
- QTableWidget * m_globalSpeakerTable
- QTableWidget * m_segmentTable
- QPushButton * m_applyButton
- QPushButton * m_okButton
- QPushButton * m_cancelButton
- QLabel * m_statusLabel
- QTimer * m_statusTimer
- QLineEdit * m_mergeNameEdit
- QPushButton * m_mergeSpeakersButton
- QSet< QString > m_allKnownSpeakers

  Eine Menge aller einzigartigen Sprechernamen im Transkript.
- QMap< QString, QString > m_currentGlobalNames

  Puffer für globale Namensänderungen.
- QMap< QPair< QString, QString >, QString > m_currentSegmentNames

  Puffer für segmentweise Sprecheränderungen.
- QString m_selectedSegmentStart

  Merker für das programmatisch ausgewählte Segment.
- QString m_selectedSegmentEnd

7.12.1 Detailed Description

Ein nicht-modaler Dialog zur Bearbeitung von Sprecherinformationen im Transkript.

Der Dialog bietet zwei Ansichten (Tabs):

1. Eine globale Ansicht, um einen Sprechernamen im gesamten Transkript zu ändern (z.B. "SPEAKER_00" in "Max Mustermann").
2. Eine segment-basierte Ansicht, um den Sprecher für einzelne Textabschnitte zu korrigieren. Der Dialog operiert auf einem ihm übergebenen Transcription-Objekt.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 SpeakerEditorDialog()

```
SpeakerEditorDialog::SpeakerEditorDialog (
            Transcription * transcription,
            QWidget * parent = nullptr )  [explicit]
```

Konstruktor, der den Dialog initialisiert und mit dem Datenmodell verknüpft.

Parameters

transcription	Ein Zeiger auf das Transcription-Objekt, das bearbeitet werden soll.
parent	Das QObject-Elternteil für die Speicherverwaltung.

7.12.3 Member Function Documentation

7.12.3.1 applyCurrentTabChanges()

```
void SpeakerEditorDialog::applyCurrentTabChanges ( )  [private]
```

Wendet die im aktuell aktiven Tab vorgenommenen Änderungen auf das Transcription-Objekt an.

7.12.3.2 handleApplyOkButtonClicked

```
void SpeakerEditorDialog::handleApplyOkButtonClicked ( )  [private], [slot]
```

Slot für die "Anwenden" und "OK" Buttons. Übernimmt die Änderungen.

7.12.3.3 handleCancelButtonClicked

```
void SpeakerEditorDialog::handleCancelButtonClicked ( )  [private], [slot]
```

Slot für den "Abbrechen" Button. Schließt den Dialog ohne Änderungen.

7.12.3.4 onGlobalSpeakerNameChanged

```
void SpeakerEditorDialog::onGlobalSpeakerNameChanged (
            const QString & text )  [private], [slot]
```

Reagiert auf eine Textänderung im Eingabefeld für einen globalen Sprechernamen.

7.12.3.5 onMergeSpeakersClicked

```
void SpeakerEditorDialog::onMergeSpeakersClicked ( )  [private], [slot]
```

Führt die ausgewählten Sprecher unter einem neuen Namen zusammen.

7.12.3.6 onSegmentSpeakerChanged

```
void SpeakerEditorDialog::onSegmentSpeakerChanged (
            int index )  [private], [slot]
```

Reagiert auf eine Änderung in der Sprecher-ComboBox eines Segments.

7.12.3.7 onTranscriptionChanged

```
void SpeakerEditorDialog::onTranscriptionChanged ( )  [slot]
```

Aktualisiert die Ansichten des Dialogs, wenn sich das zugrundeliegende Transcription-Objekt ändert.

Note

Sollte mit dem `Transcription::changed()` Signal verbunden werden.

7.12.3.8 populateGlobalSpeakerTable

```
void SpeakerEditorDialog::populateGlobalSpeakerTable ( )  [private], [slot]
```

Füllt die Tabelle für die globalen Sprechernamen mit Daten.

7.12.3.9 populateSegmentTable

```
void SpeakerEditorDialog::populateSegmentTable ( )  [private], [slot]
```

Füllt die Tabelle für die einzelnen Textsegmente mit Daten.

7.12.3.10 setDialogStatus

```
void SpeakerEditorDialog::setDialogStatus (
            const QString & text,
            bool temporary = true )  [private], [slot]
```

Zeigt eine temporäre oder permanente Nachricht in der Statuszeile des Dialogs an.

7.12.3.11 setSelectedSegment()

```
void SpeakerEditorDialog::setSelectedSegment (
            const QString & start,
            const QString & end )
```

Ermöglicht das programmatische Vor-Auswählen eines Segments im Dialog.

Parameters

start	Der Start-Zeitstempel des zu selektierenden Segments.
end	Der End-Zeitstempel des zu selektierenden Segments.

7.12.3.12 setupUI

```
void SpeakerEditorDialog::setupUI ( )  [private], [slot]
```

Erstellt die Benutzeroberfläche und verbindet die internen Signale und Slots.

7.12.3.13 updateKnownSpeakers()

```
void SpeakerEditorDialog::updateKnownSpeakers ( )  [private]
```

Aktualisiert die interne Liste aller bekannten Sprechernamen aus dem Transkript.

7.12.4 Member Data Documentation

7.12.4.1 m_allKnownSpeakers

```
QSet<QString> SpeakerEditorDialog::m_allKnownSpeakers  [private]
```

Eine Menge aller einzigartigen Sprechernamen im Transkript.

7.12.4.2 m_applyButton

```
QPushButton* SpeakerEditorDialog::m_applyButton  [private]
```

7.12.4.3 m_cancelButton

```
QPushButton* SpeakerEditorDialog::m_cancelButton  [private]
```

7.12.4.4 m_currentGlobalNames

```
QMap<QString, QString> SpeakerEditorDialog::m_currentGlobalNames  [private]
```

Puffer für globale Namensänderungen.

7.12.4.5 m_currentSegmentNames

```
QMap<QPair<QString, QString>, QString> SpeakerEditorDialog::m_currentSegmentNames  [private]
```

Puffer für segmentweise Sprecheränderungen.

7.12.4.6 m_globalSpeakerTable

```
QTableWidget* SpeakerEditorDialog::m_globalSpeakerTable  [private]
```

7.12.4.7 m_mergeNameEdit

```
QLineEdit* SpeakerEditorDialog::m_mergeNameEdit  [private]
```

7.12.4.8 m_mergeSpeakersButton

```
QPushButton* SpeakerEditorDialog::m_mergeSpeakersButton  [private]
```

7.12.4.9 m_okButton

```
QPushButton* SpeakerEditorDialog::m_okButton  [private]
```

7.12.4.10 m_segmentTable

```
QTableWidget* SpeakerEditorDialog::m_segmentTable  [private]
```

7.12.4.11 m_selectedSegmentEnd

```
QString SpeakerEditorDialog::m_selectedSegmentEnd  [private]
```

7.12.4.12 m_selectedSegmentStart

```
QString SpeakerEditorDialog::m_selectedSegmentStart  [private]
```

Merker für das programmatisch ausgewählte Segment.

7.12.4.13 m_statusLabel

```
QLabel* SpeakerEditorDialog::m_statusLabel  [private]
```

7.12.4.14 m_statusTimer

```
QTimer* SpeakerEditorDialog::m_statusTimer  [private]
```

7.12.4.15 m_tabWidget

```
QTabWidget* SpeakerEditorDialog::m_tabWidget  [private]
```

7.12.4.16 m_transcription

`QPointer<Transcription> SpeakerEditorDialog::m_transcription [private]`

Ein sicherer Zeiger auf das Transkript-Datenmodell.

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

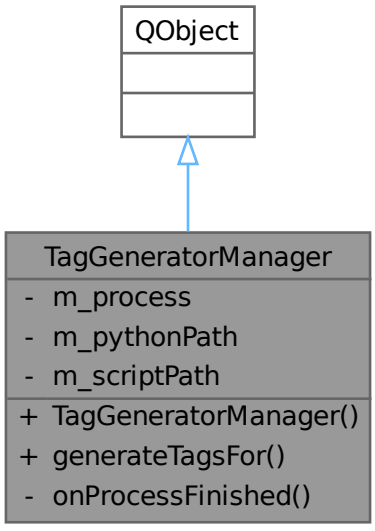
- speakereditordialog.h
- speakereditordialog.cpp

7.13 TagGeneratorManager Class Reference

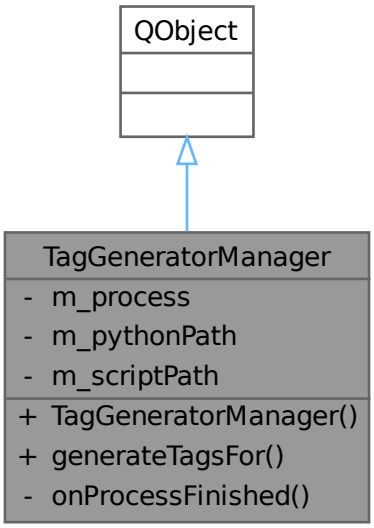
Steuert den externen Python-Prozess zur automatischen Tag-Erstellung.

`#include <taggeneratormanager.h>`

Inheritance diagram for TagGeneratorManager:



Collaboration diagram for TagGeneratorManager:



Public Slots

- void generateTagsFor (const QString &fullText)

  Startet die Tag-Analyse für den übergebenen Text.

Signals

- void tagsReady (const QStringList &tags, bool success, const QString &errorMsg="")

  Wird gesendet, wenn der Tag-Generierungs-Prozess abgeschlossen ist.

Public Member Functions

- TagGeneratorManager (QObject *parent=nullptr)

  Standard-Konstruktor.

Private Slots

- void onProcessFinished (int exitCode, QProcess::ExitStatus exitStatus)

  Interner Slot, der aufgerufen wird, wenn der Python-Prozess beendet ist.

Private Attributes

- QProcess * m_process

  Zeiger auf das QProcess-Objekt.
- QString m_pythonPath

  Pfad zum Python-Interpreter.
- QString m_scriptPath

  Pfad zum 'generate_tags.py'-Skript.

7.13.1 Detailed Description

Steuert den externen Python-Prozess zur automatischen Tag-Erstellung.

Diese Klasse nutzt spaCy über ein Python-Skript, um den Inhalt eines Transkripts zu analysieren und relevante Schlüsselwörter sowie Eigennamen (Personen, Orte, etc.) als Tags zu extrahieren. Die Ausführung geschieht asynchron in einem eigenen Prozess.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 TagGeneratorManager()

```
TagGeneratorManager::TagGeneratorManager (
            QObject * parent = nullptr )  [explicit]
```

Standard-Konstruktor.

Parameters

parent	Das QObject-Elternteil für die Speicherverwaltung.

7.13.3 Member Function Documentation

7.13.3.1 generateTagsFor

```
void TagGeneratorManager::generateTagsFor (
            const QString & fullText )  [slot]
```

Startet die Tag-Analyse für den übergebenen Text.

Der Text wird an die Standardeingabe des Python-Prozesses übergeben. Das Ergebnis wird asynchron über das `tagsReady`-Signal zurückgemeldet.

Parameters

fullText	Der gesamte Transkript-Text, der analysiert werden soll.

7.13.3.2 onProcessFinished

```
void TagGeneratorManager::onProcessFinished (
            int exitCode,
            QProcess::ExitStatus exitStatus )  [private], [slot]
```

Interner Slot, der aufgerufen wird, wenn der Python-Prozess beendet ist.

Liest die Ausgabe des Skripts, wandelt sie in eine QStringList um und emittiert das `tagsReady`-Signal.

7.13.3.3 tagsReady

```
void TagGeneratorManager::tagsReady (
            const QStringList & tags,
            bool success,
            const QString & errorMsg = "" )  [signal]
```

Wird gesendet, wenn der Tag-Generierungs-Prozess abgeschlossen ist.

Parameters

tags	Eine Liste der gefundenen Tags. Ist im Fehlerfall leer.
success	true, wenn der Prozess erfolgreich war, andernfalls false.
errorMsg	Enthält eine Fehlermeldung, falls `success` false ist.

7.13.4 Member Data Documentation

7.13.4.1 m_process

```
QProcess* TagGeneratorManager::m_process  [private]
```

Zeiger auf das QProcess-Objekt.

7.13.4.2 m_pythonPath

```
QString TagGeneratorManager::m_pythonPath  [private]
```

Pfad zum Python-Interpreter.

7.13.4.3 m_scriptPath

```
QString TagGeneratorManager::m_scriptPath  [private]
```

Pfad zum 'generate_tags.py'-Skript.

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

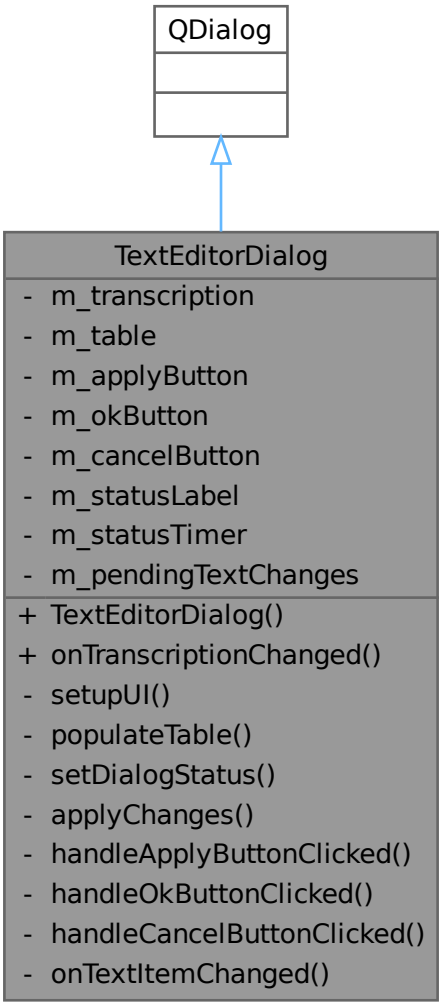
- taggeneratormanager.h
- taggeneratormanager.cpp

7.14 TextEditorDialog Class Reference

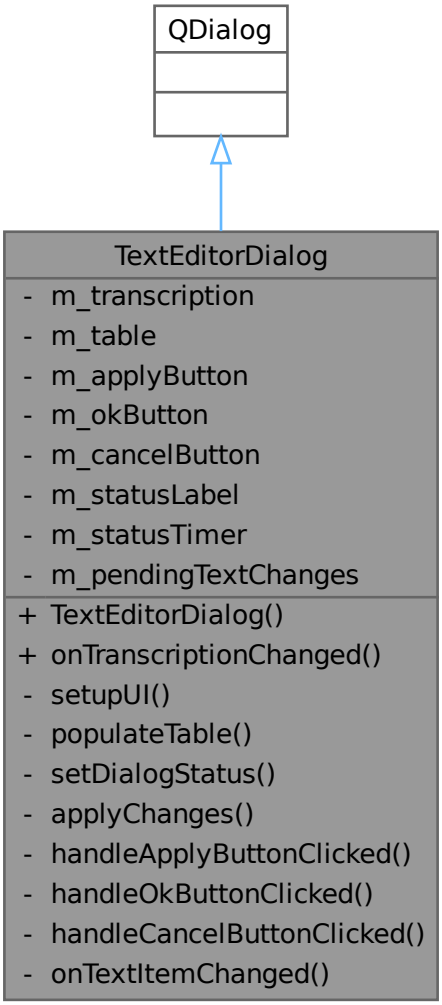
Ein nicht-modaler Dialog zur direkten Bearbeitung der Textinhalte von Transkript-Segmenten.

`#include <texteditordialog.h>`

Inheritance diagram for TextEditorDialog:



Collaboration diagram for TextEditorDialog:



Public Slots

- void onTranscriptionChanged ()

  Aktualisiert die Tabelle, falls das zugrundeliegende Transkript extern geändert wurde.

Public Member Functions

- TextEditorDialog (Transcription *transcription, QWidget *parent=nullptr)

  Konstruktor, der den Dialog initialisiert.

Private Slots

- void applyChanges ()

  Übernimmt alle gepufferten Änderungen in das Transcription-Objekt.
- void handleApplyButtonClicked ()

  Slot für den "Anwenden"-Button. Ruft applyChanges() auf.
- void handleOkButtonClicked ()

  Slot für den "OK"-Button. Ruft applyChanges() auf und schließt den Dialog.
- void handleCancelButtonClicked ()

  Slot für den "Abbrechen"-Button. Schließt den Dialog ohne Änderungen.
- void onTextItemChanged (QTableWidgetItem ∗item)

  Wird aufgerufen, wenn ein Benutzer den Text in einer Zelle der Tabelle ändert.

Private Member Functions

- void setupUI ()

  Erstellt und konfiguriert die UI-Elemente des Dialogs.
- void populateTable ()

  Füllt die Tabelle mit den Segment-Daten aus dem Transcription-Objekt.
- void setDialogStatus (const QString &text, bool temporary=true)

  Zeigt eine Statusnachricht im Dialog an.

Private Attributes

- QPointer< Transcription > m_transcription

  Sicherer Zeiger auf das Datenmodell.
- QTableWidget ∗ m_table
- QPushButton ∗ m_applyButton
- QPushButton ∗ m_okButton
- QPushButton ∗ m_cancelButton
- QLabel ∗ m_statusLabel
- QTimer ∗ m_statusTimer
- QMap< QPair< QString, QString >, QString > m_pendingTextChanges

  Puffer für noch nicht gespeicherte Textänderungen. Der Key ist ein Paar aus Start-/End-Zeitstempel, der Value ist der neue Text.

7.14.1 Detailed Description

Ein nicht-modaler Dialog zur direkten Bearbeitung der Textinhalte von Transkript-Segmenten.

Der Dialog stellt alle Segmente des Transkripts in einer Tabelle dar, in der der Text direkt editiert werden kann. Änderungen werden gepuffert und erst durch einen Klick auf "Anwenden" oder "OK" in das Transcription-Datenmodell übernommen.

7.14.2 Constructor & Destructor Documentation

7.14.2.1 TextEditorDialog()

```
TextEditorDialog::TextEditorDialog (
            Transcription * transcription,
            QWidget * parent = nullptr )  [explicit]
```

Konstruktor, der den Dialog initialisiert.

Parameters

transcription	Ein Zeiger auf das Transcription-Objekt, dessen Texte bearbeitet werden sollen.
parent	Das QObject-Elternteil für die Speicherverwaltung.

7.14.3 Member Function Documentation

7.14.3.1 applyChanges

```
void TextEditorDialog::applyChanges ( )  [private], [slot]
```

Übernimmt alle gepufferten Änderungen in das Transcription-Objekt.

7.14.3.2 handleApplyButtonClicked

```
void TextEditorDialog::handleApplyButtonClicked ( )  [private], [slot]
```

Slot für den "Anwenden"-Button. Ruft applyChanges() auf.

7.14.3.3 handleCancelButtonClicked

```
void TextEditorDialog::handleCancelButtonClicked ( )  [private], [slot]
```

Slot für den "Abbrechen"-Button. Schließt den Dialog ohne Änderungen.

7.14.3.4 handleOkButtonClicked

```
void TextEditorDialog::handleOkButtonClicked ( )  [private], [slot]
```

Slot für den "OK"-Button. Ruft applyChanges() auf und schließt den Dialog.

7.14.3.5 onTextItemChanged

```
void TextEditorDialog::onTextItemChanged (
            QTableWidgetItem * item )  [private], [slot]
```

Wird aufgerufen, wenn ein Benutzer den Text in einer Zelle der Tabelle ändert.

7.14.3.6 onTranscriptionChanged

```
void TextEditorDialog::onTranscriptionChanged ( )  [slot]
```

Aktualisiert die Tabelle, falls das zugrundeliegende Transkript extern geändert wurde.

Note

Sollte mit dem `Transcription::changed()` Signal verbunden werden.

7.14.3.7 populateTable()

```
void TextEditorDialog::populateTable ( )  [private]
```

Füllt die Tabelle mit den Segment-Daten aus dem Transcription-Objekt.

7.14.3.8 setDialogStatus()

```
void TextEditorDialog::setDialogStatus (
            const QString & text,
            bool temporary = true )  [private]
```

Zeigt eine Statusnachricht im Dialog an.

7.14.3.9 setupUI()

```
void TextEditorDialog::setupUI ( )  [private]
```

Erstellt und konfiguriert die UI-Elemente des Dialogs.

7.14.4 Member Data Documentation

7.14.4.1 m_applyButton

```
QPushButton* TextEditorDialog::m_applyButton  [private]
```

7.14.4.2 m_cancelButton

```
QPushButton* TextEditorDialog::m_cancelButton  [private]
```

7.14.4.3 m_okButton

```
QPushButton* TextEditorDialog::m_okButton  [private]
```

7.14.4.4 m_pendingTextChanges

```
QMap<QPair<QString, QString>, QString> TextEditorDialog::m_pendingTextChanges  [private]
```

Puffer für noch nicht gespeicherte Textänderungen. Der Key ist ein Paar aus Start-/End-Zeitstempel, der Value ist der neue Text.

7.14.4.5 m_statusLabel

```
QLabel* TextEditorDialog::m_statusLabel  [private]
```

7.14.4.6 m_statusTimer

`QTimer* TextEditorDialog::m_statusTimer [private]`

7.14.4.7 m_table

`QTableWidget* TextEditorDialog::m_table [private]`

7.14.4.8 m_transcription

`QPointer<Transcription> TextEditorDialog::m_transcription [private]`

Sicherer Zeiger auf das Datenmodell.

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

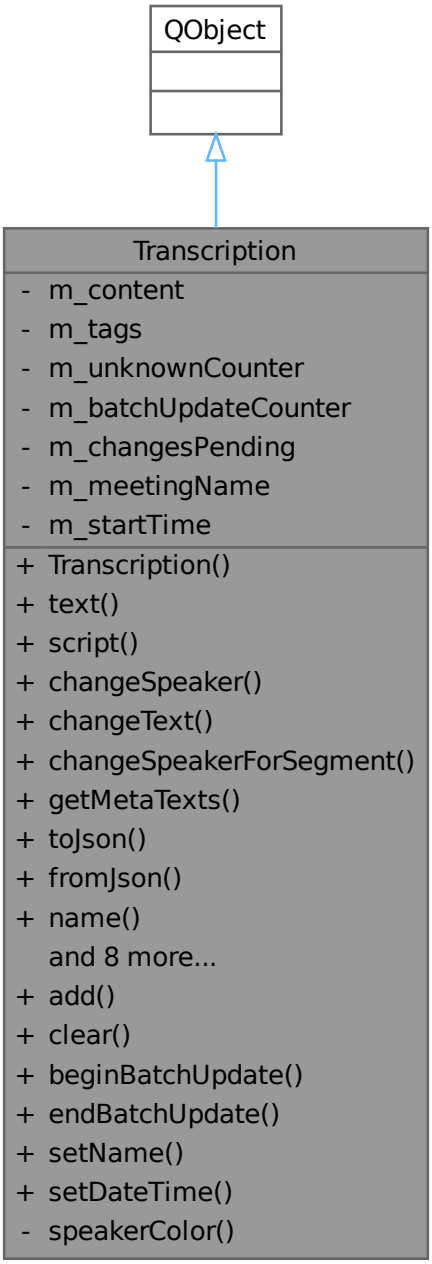
- texteditordialog.h
- texteditordialog.cpp

7.15 Transcription Class Reference

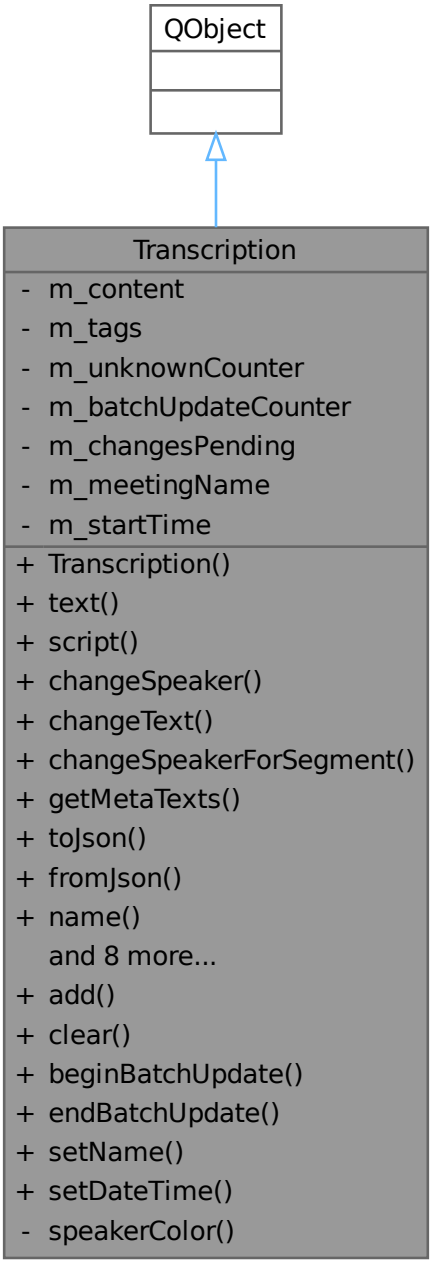
Das zentrale Datenmodell für ein komplettes Meeting-Transkript.

`#include <transcription.h>`

Inheritance diagram for Transcription:



Collaboration diagram for Transcription:



Public Slots

- void add (const MetaText &part)

  Fügt ein neues Textsegment zum Transkript hinzu.
- void clear ()

  Löscht alle Inhalte und Metadaten des Transkripts.
- void beginBatchUpdate ()

Startet einen Batch-Update-Modus, um mehrere Änderungen ohne exzessive Signal-Emissionen durchzuführen.

- void endBatchUpdate ()

  Beendet den Batch-Update-Modus und sendet ggf. ein einzelnes `changed()` -Signal.
- void setName (const QString &name)

  Setzt den Namen des Meetings.
- void setDateTime (QDateTime dateTime)

  Setzt das Startdatum und die Uhrzeit des Meetings.

Signals

- void changed ()

  Wird bei jeder sichtbaren Änderung der Daten gesendet. Dient der UI-Aktualisierung.
- void edited ()

  Wird bei jeder Änderung gesendet, die für die Undo/Redo-Funktionalität relevant ist.

Public Member Functions

- Transcription (QObject *parent=nullptr)
- QString text () const

  Gibt den reinen, zusammenhängenden Text aller Segmente zurück.
- QString script () const

  Erzeugt eine farbige HTML-Repräsentation des Transkripts für die Anzeige.
- bool changeSpeaker (const QString &oldSpeaker, const QString &newSpeaker)

  Ändert einen Sprechernamen global im gesamten Transkript.
- bool changeText (const QString &start, const QString &end, const QString &newText)

  Ändert den Text eines einzelnen, durch Zeitstempel identifizierten Segments.
- bool changeSpeakerForSegment (const QString &start, const QString &end, const QString &newSpeaker)

  Ändert den Sprecher für ein einzelnes, durch Zeitstempel identifiziertes Segment.
- const QList< MetaText > & getMetaTexts () const

  Gibt eine konstante Referenz auf die Liste aller Textsegmente zurück.
- QJsonDocument toJson () const

  Serialisiert den gesamten Zustand des Objekts in ein QJsonDocument.
- bool fromJson (const QByteArray &data)

  Füllt das Objekt mit Daten aus einem JSON-Byte-Array. Gibt bei Erfolg true zurück.
- QString name () const
- QDateTime dateTime () const
- QString getDurationAsString () const
- QStringList tags () const
- void setTags (const QStringList &tags)
- void addTag (const QString &tag)
- void removeTag (const QString &tag)
- bool hasTag (const QString &tag) const
- QList< MetaText > segmentsWithTag (const QString &tag) const

Private Member Functions

- QColor speakerColor (const QString &speaker) const

  Generiert eine deterministische Farbe basierend auf dem Sprechernamen.

Private Attributes

- QList< MetaText > m_content

  Die Liste aller transkribierten Textsegmente.
- QStringList m_tags

  Globale Tags, die für das gesamte Meeting gelten.
- int m_unknownCounter {0}

  Zähler für die Benennung anonymer Sprecher.
- int m_batchUpdateCounter {0}

  Zähler für verschachtelte Batch-Updates.
- bool m_changesPending = false

  Flag, das merkt, ob während eines Batch-Updates Änderungen aufgetreten sind.
- QString m_meetingName

  Der Name des Meetings.
- QDateTime m_startTime

  Das Startdatum und die -uhrzeit des Meetings.

7.15.1 Detailed Description

Das zentrale Datenmodell für ein komplettes Meeting-Transkript.

Diese Klasse kapselt alle Informationen für eine einzelne Aufnahmesession, inklusive der Metadaten (Meeting-↩ Name, Datum), einer Liste aller Textsegmente (`MetaText`) und globaler Tags. Sie bietet Methoden zur Bearbeitung und zur Serialisierung des gesamten Objekts nach/von JSON. Änderungen am Zustand werden über Signale (`changed`, `edited`) mitgeteilt.

7.15.2 Constructor & Destructor Documentation

7.15.2.1 Transcription()

```
Transcription::Transcription (
            QObject * parent = nullptr )  [explicit]
```

7.15.3 Member Function Documentation

7.15.3.1 add

```
void Transcription::add (
            const MetaText & part )  [slot]
```

Fügt ein neues Textsegment zum Transkript hinzu.

7.15.3.2 addTag()

```
void Transcription::addTag (
            const QString & tag )
```

7.15.3.3 beginBatchUpdate

```
void Transcription::beginBatchUpdate ( )  [slot]
```

Startet einen Batch-Update-Modus, um mehrere Änderungen ohne exzessive Signal-Emissionen durchzuführen.

7.15.3.4 changed

```
void Transcription::changed ( )  [signal]
```

Wird bei jeder sichtbaren Änderung der Daten gesendet. Dient der UI-Aktualisierung.

7.15.3.5 changeSpeaker()

```
bool Transcription::changeSpeaker (
            const QString & oldSpeaker,
            const QString & newSpeaker )
```

Ändert einen Sprechernamen global im gesamten Transkript.

7.15.3.6 changeSpeakerForSegment()

```
bool Transcription::changeSpeakerForSegment (
            const QString & start,
            const QString & end,
            const QString & newSpeaker )
```

Ändert den Sprecher für ein einzelnes, durch Zeitstempel identifiziertes Segment.

7.15.3.7 changeText()

```
bool Transcription::changeText (
            const QString & start,
            const QString & end,
            const QString & newText )
```

Ändert den Text eines einzelnen, durch Zeitstempel identifizierten Segments.

7.15.3.8 clear

```
void Transcription::clear ( )  [slot]
```

Löscht alle Inhalte und Metadaten des Transkripts.

7.15.3.9 dateTime()

```
QDateTime Transcription::dateTime ( ) const  [inline]
```

7.15.3.10 edited

```
void Transcription::edited ( )  [signal]
```

Wird bei jeder Änderung gesendet, die für die Undo/Redo-Funktionalität relevant ist.

7.15.3.11 endBatchUpdate

```
void Transcription::endBatchUpdate ( )  [slot]
```

Beendet den Batch-Update-Modus und sendet ggf. ein einzelnes `changed()`-Signal.

7.15.3.12 fromJson()

```
bool Transcription::fromJson (
            const QByteArray & data )
```

Füllt das Objekt mit Daten aus einem JSON-Byte-Array. Gibt bei Erfolg true zurück.

7.15.3.13 getDurationAsString()

```
QString Transcription::getDurationAsString ( ) const
```

7.15.3.14 getMetaTexts()

```
const QList< MetaText > & Transcription::getMetaTexts ( ) const  [inline]
```

Gibt eine konstante Referenz auf die Liste aller Textsegmente zurück.

7.15.3.15 hasTag()

```
bool Transcription::hasTag (
            const QString & tag ) const
```

7.15.3.16 name()

```
QString Transcription::name ( ) const  [inline]
```

7.15.3.17 removeTag()

```
void Transcription::removeTag (
            const QString & tag )
```

7.15.3.18 script()

```
QString Transcription::script ( ) const
```

Erzeugt eine farbige HTML-Repräsentation des Transkripts für die Anzeige.

Note

Diese Methode koppelt das Datenmodell an die Darstellung. Ein alternativer Ansatz wäre, das HTML direkt in der UI-Schicht (z.B. in MainWindow) zu generieren.

Returns

Ein HTML-formatierter QString.

7.15.3.19 segmentsWithTag()

```
QList< MetaText > Transcription::segmentsWithTag (
            const QString & tag ) const
```

7.15.3.20 setDateTime

```
void Transcription::setDateTime (
            QDateTime dateTime )  [slot]
```

Setzt das Startdatum und die Uhrzeit des Meetings.

7.15.3.21 setName

```
void Transcription::setName (
            const QString & name )  [slot]
```

Setzt den Namen des Meetings.

7.15.3.22 setTags()

```
void Transcription::setTags (
            const QStringList & tags )
```

7.15.3.23 speakerColor()

```
QColor Transcription::speakerColor (
            const QString & speaker ) const  [private]
```

Generiert eine deterministische Farbe basierend auf dem Sprechernamen.

7.15.3.24 tags()

```
QStringList Transcription::tags ( ) const  [inline]
```

7.15.3.25 text()

```
QString Transcription::text ( ) const
```

Gibt den reinen, zusammenhängenden Text aller Segmente zurück.

7.15.3.26 toJson()

```
QJsonDocument Transcription::toJson ( ) const
```

Serialisiert den gesamten Zustand des Objekts in ein QJsonDocument.

7.15.4 Member Data Documentation

7.15.4.1 m_batchUpdateCounter

```
int Transcription::m_batchUpdateCounter {0}  [private]
```

Zähler für verschachtelte Batch-Updates.

7.15.4.2 m_changesPending

```
bool Transcription::m_changesPending = false  [private]
```

Flag, das merkt, ob während eines Batch-Updates Änderungen aufgetreten sind.

7.15.4.3 m_content

```
QList<MetaText> Transcription::m_content  [private]
```

Die Liste aller transkribierten Textsegmente.

7.15.4.4 m_meetingName

```
QString Transcription::m_meetingName  [private]
```

Der Name des Meetings.

7.15.4.5 m_startTime

```
QDateTime Transcription::m_startTime  [private]
```

Das Startdatum und die -uhrzeit des Meetings.

7.15.4.6 m_tags

```
QStringList Transcription::m_tags  [private]
```

Globale Tags, die für das gesamte Meeting gelten.

7.15.4.7 m_unknownCounter

```
int Transcription::m_unknownCounter {0}  [private]
```

Zähler für die Benennung anonymer Sprecher.

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

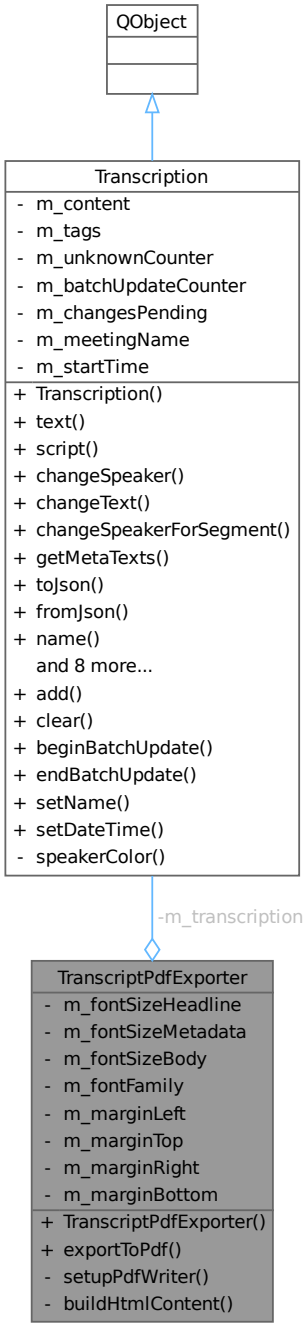
- transcription.h
- transcription.cpp

7.16 TranscriptPdfExporter Class Reference

Erstellt eine formatierte, mehrseitige PDF-Repräsentation eines Transcription-Objekts.

```
#include <transcriptpdfexporter.h>
```

Collaboration diagram for TranscriptPdfExporter:



Public Member Functions

- TranscriptPdfExporter (const Transcription &transcription)

  Konstruktor, der das zu exportierende Transkript entgegennimmt und die Layout-Einstellungen lädt.
- bool exportToPdf (const QString &filePath) const

  Führt den Export durch und speichert das Ergebnis im angegebenen Dateipfad.

Private Member Functions

- void setupPdfWriter (QPdfWriter &writer) const

  Konfiguriert das QPdfWriter-Objekt mit Seitengröße, Auflösung und Rändern.
- void buildHtmlContent (QString &html) const

  Erstellt den gesamten HTML-Code für das PDF-Dokument.

Private Attributes

- const Transcription & m_transcription

  Eine konstante Referenz auf das zu exportierende Datenmodell.
- int m_fontSizeHeadline

  Schriftgröße für die Hauptüberschrift in pt.
- int m_fontSizeMetadata

  Schriftgröße für den Metadaten-Block in pt.
- int m_fontSizeBody

  Schriftgröße für den Haupttext (Dialog) in pt.
- QString m_fontFamily

  Name der zu verwendenden Schriftfamilie (z.B. "sans-serif").
- int m_marginLeft

  Linker Seitenrand in mm.
- int m_marginTop

  Oberer Seitenrand in mm.
- int m_marginRight

  Rechter Seitenrand in mm.
- int m_marginBottom

  Unterer Seitenrand in mm.

7.16.1 Detailed Description

Erstellt eine formatierte, mehrseitige PDF-Repräsentation eines Transcription-Objekts.

Diese Klasse ist als "One-Shot"-Utility konzipiert. Man erstellt ein Objekt, das die zu exportierenden Daten und die Layout-Einstellungen (aus QSettings) einliest. Die Methode exportToPdf() generiert dann die fertige Datei.

7.16.2 Constructor & Destructor Documentation

7.16.2.1 TranscriptPdfExporter()

```
TranscriptPdfExporter::TranscriptPdfExporter (
            const Transcription & transcription )  [explicit]
```

Konstruktor, der das zu exportierende Transkript entgegennimmt und die Layout-Einstellungen lädt.

Parameters

transcription	Das Transcription-Datenmodell, das als PDF exportiert werden soll.

7.16.3 Member Function Documentation

7.16.3.1 buildHtmlContent()

```
void TranscriptPdfExporter::buildHtmlContent (
            QString & html ) const  [private]
```

Erstellt den gesamten HTML-Code für das PDF-Dokument.

7.16.3.2 exportToPdf()

```
bool TranscriptPdfExporter::exportToPdf (
            const QString & filePath ) const
```

Führt den Export durch und speichert das Ergebnis im angegebenen Dateipfad.

Diese Methode orchestriert den gesamten Prozess: Erstellen des HTML-Inhalts, Aufsetzen des QTextDocument und Aufruf der print()-Funktion, welche die Paginierung automatisch handhabt.

Parameters

filePath	Der vollständige Pfad, unter dem die PDF-Datei gespeichert werden soll.

Returns

true bei Erfolg, andernfalls false.

7.16.3.3 setupPdfWriter()

```
void TranscriptPdfExporter::setupPdfWriter (
            QPdfWriter & writer ) const  [private]
```

Konfiguriert das QPdfWriter-Objekt mit Seitengröße, Auflösung und Rändern.

7.16.4 Member Data Documentation

7.16.4.1 m_fontFamily

```
QString TranscriptPdfExporter::m_fontFamily  [private]
```

Name der zu verwendenden Schriftfamilie (z.B. "sans-serif").

7.16.4.2 m_fontSizeBody

```
int TranscriptPdfExporter::m_fontSizeBody  [private]
```

Schriftgröße für den Haupttext (Dialog) in pt.

7.16.4.3 m_fontSizeHeadline

```
int TranscriptPdfExporter::m_fontSizeHeadline  [private]
```

Schriftgröße für die Hauptüberschrift in pt.

7.16.4.4 m_fontSizeMetadata

```
int TranscriptPdfExporter::m_fontSizeMetadata  [private]
```

Schriftgröße für den Metadaten-Block in pt.

7.16.4.5 m_marginBottom

```
int TranscriptPdfExporter::m_marginBottom  [private]
```

Unterer Seitenrand in mm.

7.16.4.6 m_marginLeft

```
int TranscriptPdfExporter::m_marginLeft  [private]
```

Linker Seitenrand in mm.

7.16.4.7 m_marginRight

```
int TranscriptPdfExporter::m_marginRight  [private]
```

Rechter Seitenrand in mm.

7.16.4.8 m_marginTop

```
int TranscriptPdfExporter::m_marginTop  [private]
```

Oberer Seitenrand in mm.

7.16.4.9 m_transcription

```
const Transcription& TranscriptPdfExporter::m_transcription  [private]
```

Eine konstante Referenz auf das zu exportierende Datenmodell.

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

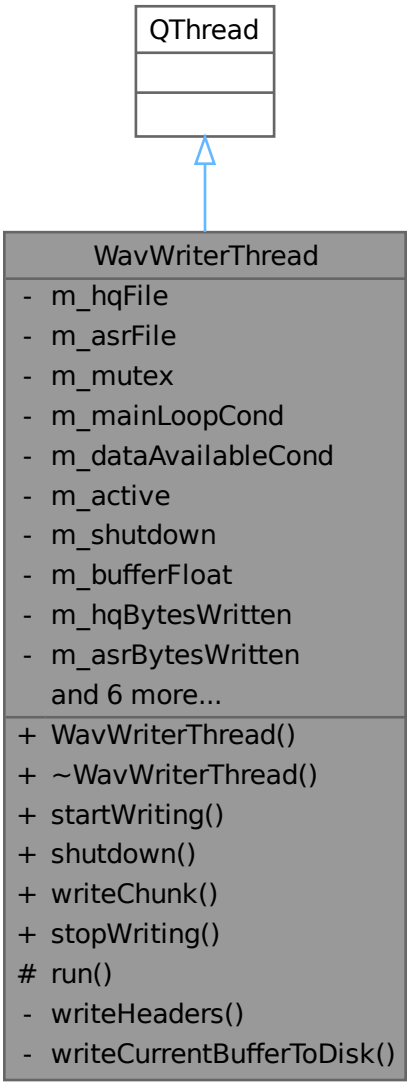
- transcriptpdfexporter.h
- transcriptpdfexporter.cpp

7.17 WavWriterThread Class Reference

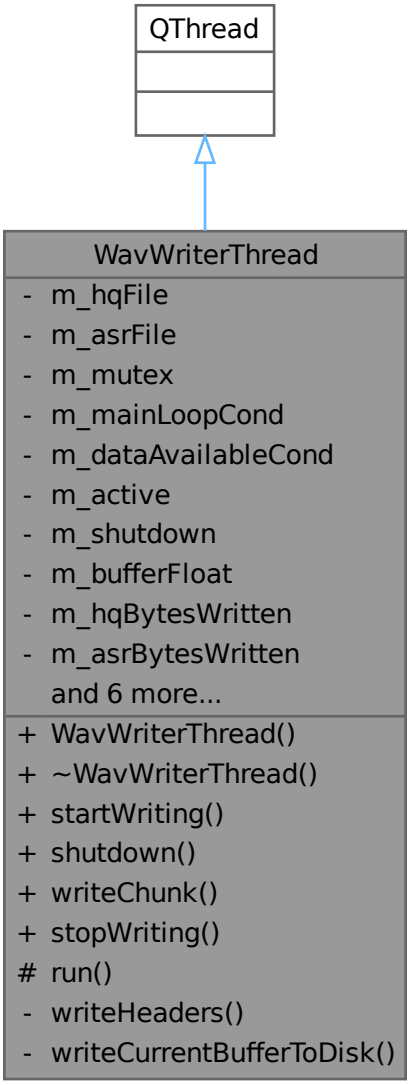
Ein dedizierter Thread, der Audio-Daten in WAV-Dateien schreibt.

`#include <wavwriterthread.h>`

Inheritance diagram for WavWriterThread:



Collaboration diagram for WavWriterThread:



Public Slots

- void writeChunk (QList< float > chunk)

  Nimmt einen Block von Audio-Daten entgegen und fügt ihn dem internen Puffer hinzu.
- void stopWriting ()

  Beendet die aktuelle Schreib-Session.

Signals

- void finishedWriting ()

  Wird gesendet, nachdem eine Schreib-Session vollständig abgeschlossen und die Dateien geschlossen wurden.

Public Member Functions

- WavWriterThread (QObject *parent=nullptr)

  Standard-Konstruktor.
- ~WavWriterThread ()

  Destruktor. Ruft shutdown() auf, um den Thread sicher zu beenden.
- void startWriting (const QString &hqPath, const QString &asrPath)

  Startet eine neue Schreib-Session.
- void shutdown ()

  Beendet den Thread vollständig und wartet auf dessen Terminierung.

Protected Member Functions

- void run () override

  Die Hauptfunktion des Threads (der "Consumer"-Teil).

Private Member Functions

- void writeHeaders (qint64 hqBytes, qint64 asrBytes)

  Schreibt die finalen WAV-Header in die Dateien, nachdem alle Daten geschrieben wurden.
- void writeCurrentBufferToDisk (QList< float > &buffer)

  Schreibt den aktuellen Inhalt des internen Puffers auf die Festplatte.

Private Attributes

- QFile m_hqFile

  Dateihandle für die High-Quality-WAV-Datei.
- QFile m_asrFile

  Dateihandle für die ASR-WAV-Datei.
- QMutex m_mutex
- QWaitCondition m_mainLoopCond

  Weckt den Thread auf, wenn startWriting() gerufen wird.
- QWaitCondition m_dataAvailableCond

  Weckt den Thread auf, wenn neue Daten in writeChunk() ankommen.
- std::atomic< bool > m_active

  Steuert die aktive Schreibschleife.
- std::atomic< bool > m_shutdown

  Signalisiert dem Thread, sich komplett zu beenden.
- QList< float > m_bufferFloat

  Interner Puffer für ankommende Audio-Chunks.
- qint64 m_hqBytesWritten

  Zähler für geschriebene Bytes (HQ).
- qint64 m_asrBytesWritten

  Zähler für geschriebene Bytes (ASR).
- qint64 m_flushThresholdBytes

  Pufferschwelle in Bytes, bevor auf die Platte geschrieben wird.
- int m_downsampleOffset

  Offset für das Downsampling zur ASR-Version.
- const int m_sampleRateHQ

Sample-Rate für die HQ-Aufnahme (z.B. 48000 Hz).

- const int m_channelsHQ

  Anzahl der Kanäle für die HQ-Aufnahme (z.B. 2 für Stereo).

- const int m_bitsPerSampleHQ

  Bittiefe für die HQ-Aufnahme (z.B. 32 für float).

- const int m_sampleRateASR

  Ziel-Sample-Rate für die ASR-Aufnahme (z.B. 16000 Hz).

7.17.1 Detailed Description

Ein dedizierter Thread, der Audio-Daten in WAV-Dateien schreibt.

Diese Klasse wird verwendet, um das Schreiben von Dateien von anderen Threads (insbesondere dem Haupt-↩ Thread und dem Aufnahme-Thread) zu entkoppeln. Sie nimmt über den `writeChunk`-Slot Audiodaten entgegen und schreibt diese in ihrem eigenen Thread-Kontext auf die Festplatte. Sie erzeugt parallel zwei Dateien: eine hochauflösende Stereo-Datei und eine für die Spracherkennung (ASR) optimierte, heruntergesampelte Mono-↩ Datei.

7.17.2 Constructor & Destructor Documentation

7.17.2.1 WavWriterThread()

```
WavWriterThread::WavWriterThread (
            QObject * parent = nullptr )  [explicit]
```

Standard-Konstruktor.

Parameters

parent	Das QObject-Elternteil für die Speicherverwaltung.

7.17.2.2 ~WavWriterThread()

```
WavWriterThread::~WavWriterThread ( )
```

Destruktor. Ruft shutdown() auf, um den Thread sicher zu beenden.

7.17.3 Member Function Documentation

7.17.3.1 finishedWriting

```
void WavWriterThread::finishedWriting ( )  [signal]
```

Wird gesendet, nachdem eine Schreib-Session vollständig abgeschlossen und die Dateien geschlossen wurden.

7.17.3.2 run()

```
void WavWriterThread::run ( )  [override], [protected]
```

Die Hauptfunktion des Threads (der "Consumer"-Teil).

Implementiert die Logik zum Warten auf Daten, Schreiben der Daten auf die Festplatte und Finalisieren der Dateien.

7.17.3.3 shutdown()

```
void WavWriterThread::shutdown ( )
```

Beendet den Thread vollständig und wartet auf dessen Terminierung.

7.17.3.4 startWriting()

```
void WavWriterThread::startWriting (
            const QString & hqPath,
            const QString & asrPath )
```

Startet eine neue Schreib-Session.

Öffnet die beiden Zieldateien und bereitet den Thread auf das Empfangen von Audio-Daten vor.

Parameters

hqPath	Pfad für die hochauflösende WAV-Datei.
asrPath	Pfad für die ASR-optimierte WAV-Datei.

7.17.3.5 stopWriting

```
void WavWriterThread::stopWriting ( )  [slot]
```

Beendet die aktuelle Schreib-Session.

Veranlasst den Thread, alle verbleibenden Daten aus dem Puffer zu schreiben, die WAV-Header zu finalisieren und die Dateien zu schließen.

7.17.3.6 writeChunk

```
void WavWriterThread::writeChunk (
            QList< float > chunk )  [slot]
```

Nimmt einen Block von Audio-Daten entgegen und fügt ihn dem internen Puffer hinzu.

Note

Dieser Slot ist thread-sicher und dazu gedacht, mit dem `pcmChunkReady`-Signal des CaptureThread verbunden zu werden.

Parameters

chunk	Ein Block von Audio-Daten (2 Kanäle, 32-bit float).

7.17.3.7 writeCurrentBufferToDisk()

```
void WavWriterThread::writeCurrentBufferToDisk (
            QList< float > & buffer )  [private]
```

Schreibt den aktuellen Inhalt des internen Puffers auf die Festplatte.

Parameters

buffer	Der Puffer, dessen Inhalt geschrieben werden soll.

7.17.3.8 writeHeaders()

```
void WavWriterThread::writeHeaders (
            qint64 hqBytes,
            qint64 asrBytes )  [private]
```

Schreibt die finalen WAV-Header in die Dateien, nachdem alle Daten geschrieben wurden.

Parameters

hqBytes	Die Gesamtgröße der geschriebenen Audiodaten für die HQ-Datei.
asrBytes	Die Gesamtgröße der geschriebenen Audiodaten für die ASR-Datei.

7.17.4 Member Data Documentation

7.17.4.1 m_active

```
std::atomic<bool> WavWriterThread::m_active  [private]
```

Steuert die aktive Schreibschleife.

7.17.4.2 m_asrBytesWritten

```
qint64 WavWriterThread::m_asrBytesWritten  [private]
```

Zähler für geschriebene Bytes (ASR).

7.17.4.3 m_asrFile

```
QFile WavWriterThread::m_asrFile  [private]
```

Dateihandle für die ASR-WAV-Datei.

7.17.4.4 m_bitsPerSampleHQ

```
const int WavWriterThread::m_bitsPerSampleHQ  [private]
```

Bittiefe für die HQ-Aufnahme (z.B. 32 für float).

7.17.4.5 m_bufferFloat

```
QList<float> WavWriterThread::m_bufferFloat  [private]
```

Interner Puffer für ankommende Audio-Chunks.

7.17.4.6 m_channelsHQ

```
const int WavWriterThread::m_channelsHQ  [private]
```

Anzahl der Kanäle für die HQ-Aufnahme (z.B. 2 für Stereo).

7.17.4.7 m_dataAvailableCond

```
QWaitCondition WavWriterThread::m_dataAvailableCond  [private]
```

Weckt den Thread auf, wenn neue Daten in writeChunk() ankommen.

7.17.4.8 m_downsampleOffset

```
int WavWriterThread::m_downsampleOffset  [private]
```

Offset für das Downsampling zur ASR-Version.

7.17.4.9 m_flushThresholdBytes

```
qint64 WavWriterThread::m_flushThresholdBytes  [private]
```

Pufferschwelle in Bytes, bevor auf die Platte geschrieben wird.

7.17.4.10 m_hqBytesWritten

```
qint64 WavWriterThread::m_hqBytesWritten  [private]
```

Zähler für geschriebene Bytes (HQ).

7.17.4.11 m_hqFile

```
QFile WavWriterThread::m_hqFile  [private]
```

Dateihandle für die High-Quality-WAV-Datei.

7.17.4.12 m_mainLoopCond

```
QWaitCondition WavWriterThread::m_mainLoopCond  [private]
```

Weckt den Thread auf, wenn startWriting() gerufen wird.

7.17.4.13 m_mutex

```
QMutex WavWriterThread::m_mutex  [private]
```

7.17.4.14 m_sampleRateASR

```
const int WavWriterThread::m_sampleRateASR  [private]
```

Ziel-Sample-Rate für die ASR-Aufnahme (z.B. 16000 Hz).

7.17.4.15 m_sampleRateHQ

```
const int WavWriterThread::m_sampleRateHQ  [private]
```

Sample-Rate für die HQ-Aufnahme (z.B. 48000 Hz).

7.17.4.16 m_shutdown

```
std::atomic<bool> WavWriterThread::m_shutdown  [private]
```

Signalisiert dem Thread, sich komplett zu beenden.

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

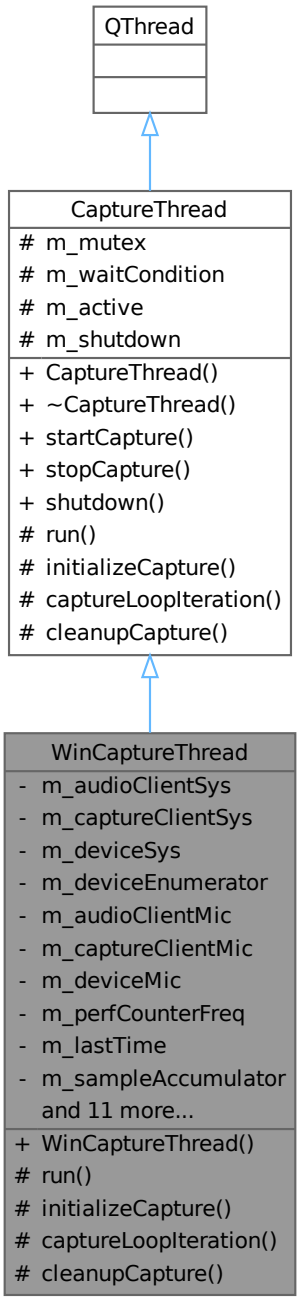
- wavwriterthread.h
- wavwriterthread.cpp

7.18 WinCaptureThread Class Reference

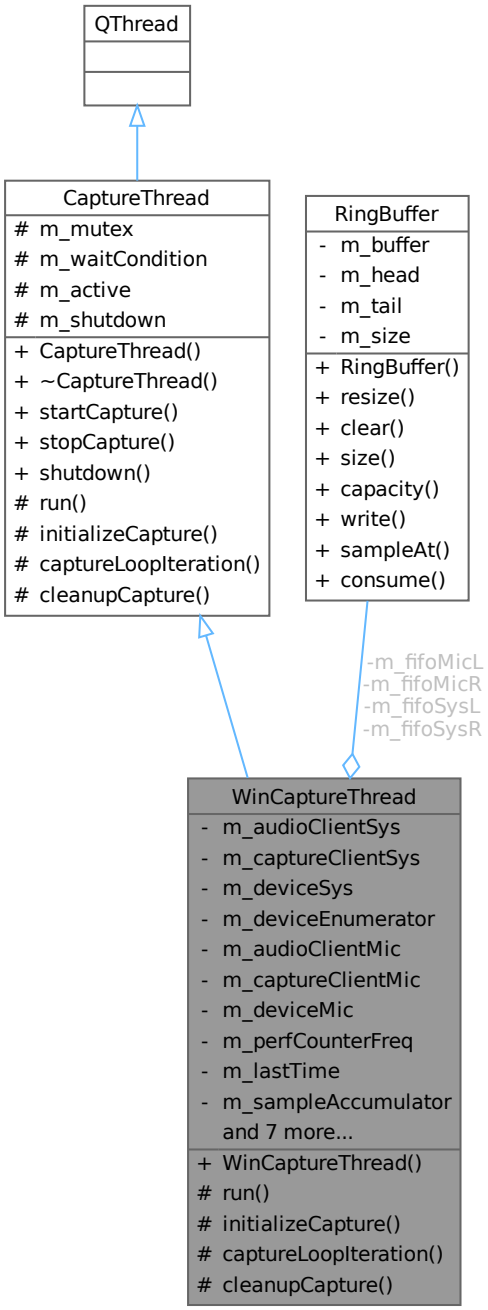
Eine konkrete Implementierung von CaptureThread für Windows-Systeme.

`#include <wincapturethread.h>`

Inheritance diagram for WinCaptureThread:



Collaboration diagram for WinCaptureThread:



Public Member Functions

- WinCaptureThread (QObject *parent=nullptr)

  Standard-Konstruktor.

Public Member Functions inherited from CaptureThread

- CaptureThread (QObject *parent=nullptr)

Standard-Konstruktor.

- virtual ~CaptureThread ()=default

  Virtueller Destruktor.

- void startCapture ()

  Startet eine neue Aufnahme-Session. Diese Methode ist thread-sicher und weckt den run()-Loop auf, um mit der Aufnahme zu beginnen.

- virtual void stopCapture ()

  Fordert das Beenden der aktuellen Aufnahme-Session an. Dies ist ein nicht-blockierender Aufruf. Der Thread beendet die Aufnahmeschleife so bald wie möglich.

- void shutdown ()

  Beendet den Thread vollständig und wartet auf dessen Terminierung. Dies ist ein blockierender Aufruf, der sicherstellt, dass alle Ressourcen freigegeben werden.

Protected Member Functions

- void run () override

  Überschriebene Hauptfunktion des Threads zur Initialisierung von COM.

- bool initializeCapture () override

  Initialisiert die WASAPI-Audio-Clients für System- und Mikrofon-Aufnahme.

- void captureLoopIteration () override

  Führt eine einzelne Iteration der Aufnahmeschleife aus.

- void cleanupCapture () override

  Gibt alle WASAPI- und COM-Ressourcen frei.

Protected Member Functions inherited from CaptureThread

- void run () override

  Die Hauptfunktion des Threads, die von QThread aufgerufen wird. Sie implementiert die Zustandsmaschine für den Aufnahme-Lebenszyklus.

Private Attributes

- IAudioClient * m_audioClientSys = nullptr

  Der WASAPI-Client für das Ausgabe-Gerät.

- IAudioCaptureClient * m_captureClientSys = nullptr

  Der Client zum Abgreifen der Loopback-Daten.

- IMMDevice * m_deviceSys = nullptr

  Repräsentiert das Standard-Ausgabegerät (z.B. Lautsprecher).

- IMMDeviceEnumerator * m_deviceEnumerator = nullptr

  Wird zur Auflistung und Auswahl von Audio-Geräten verwendet.

- IAudioClient * m_audioClientMic = nullptr

  Der WASAPI-Client für das Eingabe-Gerät.

- IAudioCaptureClient * m_captureClientMic = nullptr

  Der Client zum Abgreifen der Mikrofon-Daten.

- IMMDevice * m_deviceMic = nullptr

  Repräsentiert das Standard-Eingabegerät (Mikrofon).

- LARGE_INTEGER m_perfCounterFreq

  Frequenz des High-Performance Timers für präzises Timing.

- LARGE_INTEGER m_lastTime

  Letzter Zeitstempel für die Delta-Zeit-Berechnung.

- double m_sampleAccumulator = 0.0

  Akkumulator für eine fließkomma-genaue Frame-Zählung beim Resampling.
- double m_resampPosMic = 0.0

  Aktuelle Leseposition im Mikrofon-Ringpuffer.
- double m_resampPosSys = 0.0

  Aktuelle Leseposition im System-Audio-Ringpuffer.
- const int m_pollingIntervalMs = 10

  Das Intervall in ms, in dem neue Audiodaten abgefragt werden.
- RingBuffer m_fifoSysL

  Ringpuffer für den linken Kanal des System-Audios.
- RingBuffer m_fifoMicL

  Ringpuffer für den linken Kanal des Mikrofons.
- RingBuffer m_fifoSysR

  Ringpuffer für den rechten Kanal des System-Audios.
- RingBuffer m_fifoMicR

  Ringpuffer für den rechten Kanal des Mikrofons.
- UINT32 m_nativeSampleRateSys = 0

  Native Abtastrate des System-Audio-Geräts.
- UINT32 m_nativeChannelsSys = 0

  Native Kanalanzahl des System-Audio-Geräts.
- UINT32 m_nativeSampleRateMic = 0

  Native Abtastrate des Mikrofons.
- UINT32 m_nativeChannelsMic = 0

  Native Kanalanzahl des Mikrofons.

Additional Inherited Members

Signals inherited from CaptureThread

- void pcmChunkReady (QList< float > chunk)

  Wird gesendet, wenn ein neuer Block von Audio-Daten (PCM-Samples) bereitsteht.
- void started ()

  Wird gesendet, unmittelbar nachdem die plattformspezifische Initialisierung erfolgreich war und die Aufnahmeschleife beginnt.
- void stopped ()

  Wird gesendet, nachdem die Aufnahmeschleife beendet und die Aufräumarbeiten abgeschlossen sind.

Protected Attributes inherited from CaptureThread

- QMutex m_mutex

  Schützt den Zugriff auf den Zustand des Threads.
- QWaitCondition m_waitCondition

  Lässt den Thread schlafen, wenn er inaktiv ist.
- std::atomic< bool > m_active

  Steuert die innere Aufnahmeschleife (start/stop).
- std::atomic< bool > m_shutdown

  Signalisiert dem Thread, sich komplett zu beenden.

7.18.1 Detailed Description

Eine konkrete Implementierung von CaptureThread für Windows-Systeme.

Diese Klasse nutzt die Windows Audio Session API (WASAPI) zur Aufnahme von Audio. Sie initialisiert zwei separate Audio-Clients: einen für das Standard-Ausgabegerät im Loopback-Modus (um System-Sounds aufzunehmen) und einen für das Standard-Eingabegerät (Mikrofon). Die Daten beider Streams werden in Ringpuffern zwischengespeichert, resampelt, gemischt und dann zur weiteren Verarbeitung gesendet.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 WinCaptureThread()

```
WinCaptureThread::WinCaptureThread (
            QObject * parent = nullptr )  [explicit]
```

Standard-Konstruktor.

Parameters

parent	Das QObject-Elternteil für die Speicherverwaltung.

7.18.3 Member Function Documentation

7.18.3.1 captureLoopIteration()

```
void WinCaptureThread::captureLoopIteration ( )  [override], [protected], [virtual]
```

Führt eine einzelne Iteration der Aufnahmeschleife aus.

Liest verfügbare Datenpakete von beiden WASAPI-Clients, schreibt sie in die Ringpuffer, führt ein Resampling auf 48kHz durch und mischt die Streams.

Implements CaptureThread.

7.18.3.2 cleanupCapture()

```
void WinCaptureThread::cleanupCapture ( )  [override], [protected], [virtual]
```

Gibt alle WASAPI- und COM-Ressourcen frei.

Stoppt die Audio-Clients und gibt die Referenzen auf alle COM-Interfaces frei.

Implements CaptureThread.

7.18.3.3 initializeCapture()

```
bool WinCaptureThread::initializeCapture ( )  [override], [protected], [virtual]
```

Initialisiert die WASAPI-Audio-Clients für System- und Mikrofon-Aufnahme.

Returns

true bei Erfolg, andernfalls false.

Implements CaptureThread.

7.18.3.4 run()

```
void WinCaptureThread::run ( )  [override], [protected]
```

Überschriebene Hauptfunktion des Threads zur Initialisierung von COM.

Note

Da WASAPI auf COM basiert, muss für diesen Thread explizit die COM-Bibliothek initialisiert (CoInitializeEx) und wieder freigegeben (CoUninitialize) werden.

7.18.4 Member Data Documentation

7.18.4.1 m_audioClientMic

```
IAudioClient* WinCaptureThread::m_audioClientMic = nullptr  [private]
```

Der WASAPI-Client für das Eingabe-Gerät.

7.18.4.2 m_audioClientSys

```
IAudioClient* WinCaptureThread::m_audioClientSys = nullptr  [private]
```

Der WASAPI-Client für das Ausgabe-Gerät.

7.18.4.3 m_captureClientMic

```
IAudioCaptureClient* WinCaptureThread::m_captureClientMic = nullptr  [private]
```

Der Client zum Abgreifen der Mikrofon-Daten.

7.18.4.4 m_captureClientSys

```
IAudioCaptureClient* WinCaptureThread::m_captureClientSys = nullptr  [private]
```

Der Client zum Abgreifen der Loopback-Daten.

7.18.4.5 m_deviceEnumerator

```
IMMDeviceEnumerator* WinCaptureThread::m_deviceEnumerator = nullptr  [private]
```

Wird zur Auflistung und Auswahl von Audio-Geräten verwendet.

7.18.4.6 m_deviceMic

```
IMMDevice* WinCaptureThread::m_deviceMic = nullptr  [private]
```

Repräsentiert das Standard-Eingabegerät (Mikrofon).

7.18.4.7 m_deviceSys

```
IMMDevice* WinCaptureThread::m_deviceSys = nullptr  [private]
```

Repräsentiert das Standard-Ausgabegerät (z.B. Lautsprecher).

7.18.4.8 m_fifoMicL

```
RingBuffer WinCaptureThread::m_fifoMicL  [private]
```

Ringpuffer für den linken Kanal des Mikrofons.

7.18.4.9 m_fifoMicR

```
RingBuffer WinCaptureThread::m_fifoMicR  [private]
```

Ringpuffer für den rechten Kanal des Mikrofons.

7.18.4.10 m_fifoSysL

```
RingBuffer WinCaptureThread::m_fifoSysL  [private]
```

Ringpuffer für den linken Kanal des System-Audios.

7.18.4.11 m_fifoSysR

```
RingBuffer WinCaptureThread::m_fifoSysR  [private]
```

Ringpuffer für den rechten Kanal des System-Audios.

7.18.4.12 m_lastTime

```
LARGE_INTEGER WinCaptureThread::m_lastTime  [private]
```

Letzter Zeitstempel für die Delta-Zeit-Berechnung.

7.18.4.13 m_nativeChannelsMic

```
UINT32 WinCaptureThread::m_nativeChannelsMic = 0  [private]
```

Native Kanalanzahl des Mikrofons.

7.18.4.14 m_nativeChannelsSys

```
UINT32 WinCaptureThread::m_nativeChannelsSys = 0  [private]
```

Native Kanalanzahl des System-Audio-Geräts.

7.18.4.15 m_nativeSampleRateMic

```
UINT32 WinCaptureThread::m_nativeSampleRateMic = 0  [private]
```

Native Abtastrate des Mikrofons.

7.18.4.16 m_nativeSampleRateSys

```
UINT32 WinCaptureThread::m_nativeSampleRateSys = 0  [private]
```

Native Abtastrate des System-Audio-Geräts.

7.18.4.17 m_perfCounterFreq

```
LARGE_INTEGER WinCaptureThread::m_perfCounterFreq  [private]
```

Frequenz des High-Performance Timers für präzises Timing.

7.18.4.18 m_pollingIntervalMs

```
const int WinCaptureThread::m_pollingIntervalMs = 10  [private]
```

Das Intervall in ms, in dem neue Audiodaten abgefragt werden.

7.18.4.19 m_resampPosMic

```
double WinCaptureThread::m_resampPosMic = 0.0  [private]
```

Aktuelle Leseposition im Mikrofon-Ringpuffer.

7.18.4.20 m_resampPosSys

```
double WinCaptureThread::m_resampPosSys = 0.0  [private]
```

Aktuelle Leseposition im System-Audio-Ringpuffer.

7.18.4.21 m_sampleAccumulator

```
double WinCaptureThread::m_sampleAccumulator = 0.0  [private]
```

Akkumulator für eine fließkomma-genaue Frame-Zählung beim Resampling.

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

- wincapturethread.h
- wincapturethread.cpp

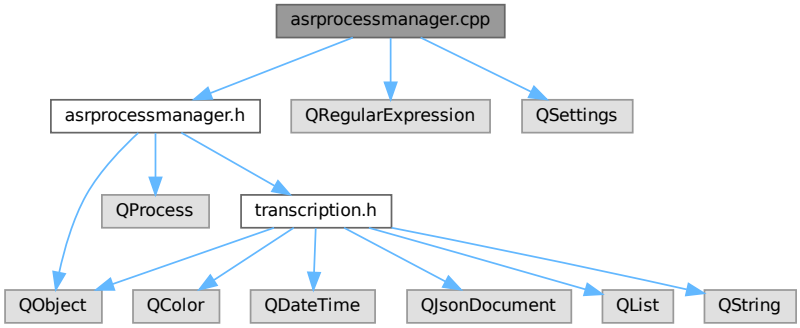
Chapter 8

File Documentation

8.1 asrprocessmanager.cpp File Reference

```
#include "asrprocessmanager.h"
#include <QRegularExpression>
#include <QSettings>
```

Include dependency graph for asrprocessmanager.cpp:



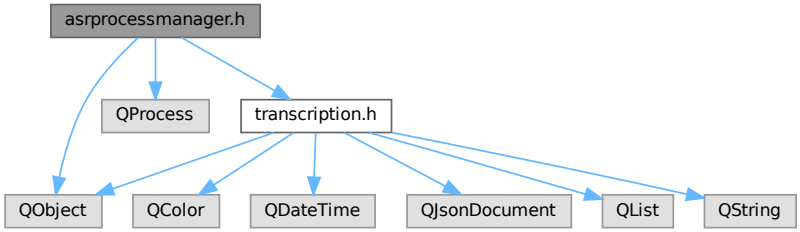
8.2 asrprocessmanager.h File Reference

Enthält die Deklaration der AsrProcessManager-Klasse.

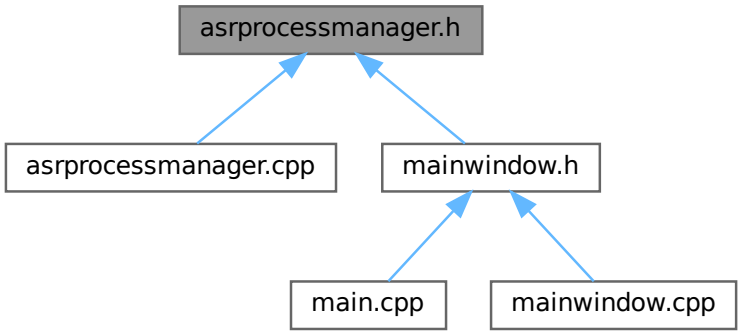
```
#include <QObject>
#include <QProcess>
```

```
#include "transcription.h"
```

Include dependency graph for asrprocessmanager.h:



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



Classes

- class AsrProcessManager

  Steuert den externen Python-Prozess für die Spracherkennung (ASR).

8.2.1 Detailed Description

Enthält die Deklaration der AsrProcessManager-Klasse.

Author

Mike Wild

8.3 asrprocessmanager.h

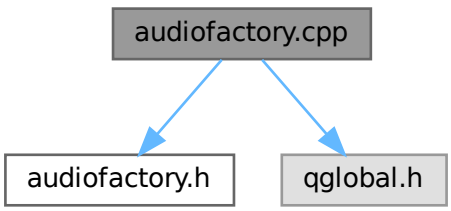
Go to the documentation of this file.

```
00001
00006 #ifndef ASRPROCESSMANAGER_H
00007 #define ASRPROCESSMANAGER_H
00008
00009 #include <QObject>
00010 #include <QProcess>
00011 #include "transcription.h"
00012
00021 class AsrProcessManager : public QObject
00022 {
00023     Q_OBJECT
00024 public:
00029     explicit AsrProcessManager (QObject *parent = nullptr);
00030
00034     ~AsrProcessManager ();
00035
00036 public slots:
00045     void startTranscription (const QString &wavFilePath);
00046
00053     void stop ();
00054
00055 signals:
00063     void segmentReady (const MetaText &segment);
00064
00071     void finished (bool success, const QString &errorMsg);
00072
00073 private slots:
00074     // Interne Slots zur Behandlung der Signale von QProcess
00076     void handleProcessOutput ();
00077
00079     void handleProcessFinished (int exitCode, QProcess::ExitStatus exitStatus);
00080
00082     void handleProcessError (QProcess::ProcessError error);
00083
00084 private:
00090     MetaText parseLine (const QString &line);
00091
00095     void loadPaths ();
00096
00097     QProcess *m_process;
00098     QString m_pythonPath;
00099     QString m_scriptPath;
00100     int m_unknownCounter;
00101 };
00102
00103 #endif // ASRPROCESSMANAGER_H
```

8.4 audiofactory.cpp File Reference

```
#include "audiofactory.h"
#include <qglobal.h>
```

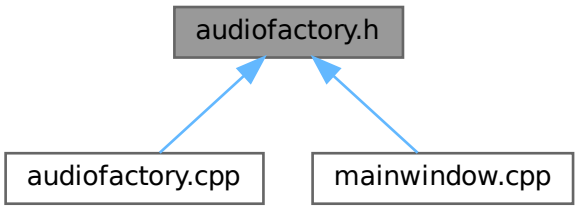
Include dependency graph for audiofactory.cpp:



8.5 audiofactory.h File Reference

Enthält die Deklaration der AudioFactory-Klasse.

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



Classes

- class AudioFactory

  Eine Factory-Klasse zur Erstellung von plattformspezifischen CaptureThread-Objekten.

8.5.1 Detailed Description

Enthält die Deklaration der AudioFactory-Klasse.

Author

Mike Wild

8.6 audiofactory.h

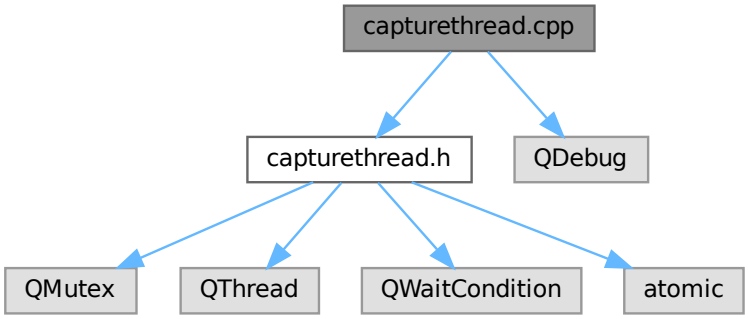
Go to the documentation of this file.

```
00001
00006 #ifndef AUDIOFACTORY_H
00007 #define AUDIOFACTORY_H
00008
00009 // Forward-Deklarationen
00010 class QObject;
00011 class CaptureThread;
00012
00020 class AudioFactory
00021 {
00022 public:
00026     AudioFactory () = default;
00027
00036     static CaptureThread *createThread (QObject *parent = nullptr);
00037 };
00038
00039 #endif // AUDIOFACTORY_H
```

8.7 capturethread.cpp File Reference

```
#include "capturethread.h"
#include <QDebug>
```

Include dependency graph for capturethread.cpp:

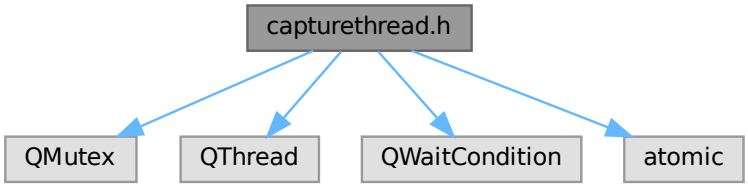


8.8 capturethread.h File Reference

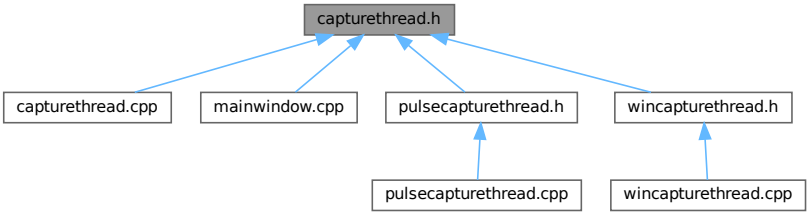
Enthält die Deklaration der abstrakten Basisklasse CaptureThread.

```
#include <QMutex>
#include <QThread>
#include <QWaitCondition>
#include <atomic>
```

Include dependency graph for capturethread.h:



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



Classes

- class CaptureThread

  Eine abstrakte Basisklasse für Threads, die Audio in Echtzeit aufnehmen.

8.8.1 Detailed Description

Enthält die Deklaration der abstrakten Basisklasse CaptureThread.

Author

Mike Wild

8.9 capturethread.h

Go to the documentation of this file.

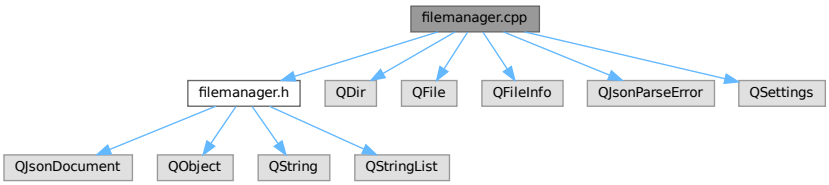
```
00001
00006 #ifndef CAPTURETHREAD_H
00007 #define CAPTURETHREAD_H
00008
00009 #include <QMutex>
00010 #include <QThread>
00011 #include <QWaitCondition>
00012 #include <atomic>
00013
00023 class CaptureThread : public QThread
00024 {
00025     Q_OBJECT
00026 public:
00031     explicit CaptureThread (QObject *parent = nullptr);
00032
00036     virtual ~CaptureThread () = default;
00037
00042     void startCapture ();
00043
00049     virtual void stopCapture ();
00050
00055     void shutdown ();
00056
00057 signals:
00062     void pcmChunkReady (QList<float> chunk);
00063
00068     void started ();
00069
00074     void stopped ();
00075
00076 protected:
00081     void run () override;
00082
```

```
00088     virtual bool initializeCapture () = 0;
00089
00095     virtual void captureLoopIteration () = 0;
00096
00102     virtual void cleanupCapture () = 0;
00103
00104     // Synchronisationsobjekte für die Steuerung des Threads von außen
00105     QMutex m_mutex;
00106     QWaitCondition m_waitCondition;
00107     std::atomic<bool> m_active;
00108     std::atomic<bool> m_shutdown;
00109 };
00110
00111 #endif // CAPTURETHREAD_H
```

8.10 filemanager.cpp File Reference

```
#include "filemanager.h"
#include <QDir>
#include <QFile>
#include <QFileInfo>
#include <QJsonParseError>
#include <QSettings>
```

Include dependency graph for filemanager.cpp:

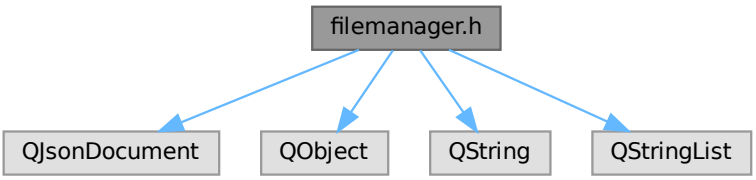


8.11 filemanager.h File Reference

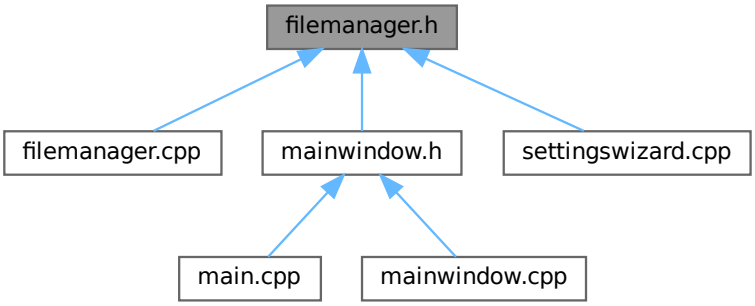
Enthält die Deklaration der FileManager-Klasse.

```
#include <QJsonDocument>
#include <QObject>
#include <QString>
#include <QStringList>
```

Include dependency graph for filemanager.h:



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



Classes

- class FileManager

  Kapselt alle direkten Dateisystem-Interaktionen der Anwendung.

8.11.1 Detailed Description

Enthält die Deklaration der FileManager-Klasse.

Author

Mike Wild

8.12 filemanager.h

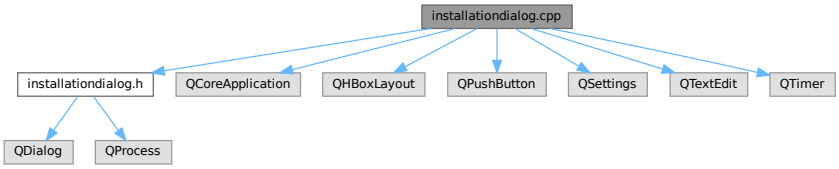
Go to the documentation of this file.

```
00001
00006 #ifndef FILEMANAGER_H
00007 #define FILEMANAGER_H
00008
00009 #include <QJsonDocument>
00010 #include <QObject>
00011 #include <QString>
00012 #include <QStringList>
00013
00022 class FileManager : public QObject
00023 {
00024     Q_OBJECT
00025 public:
00030     explicit FileManager (QObject *parent = nullptr);
00031
00037     QString getTempWavPath (bool forAsr = false) const;
00038
00046     QString getMeetingsDirectory () const;
00047
00057     QString getMeetingJsonPath (const QString &meetingId, bool isEdited) const;
00058
00065     QStringList findExistingMeetings () const;
00066
00075     QJsonDocument loadJson (const QString &filePath, bool &ok) const;
00076
00085     bool saveJson (const QString &filePath, const QJsonDocument &doc) const;
00086 };
00087
00088 #endif // FILEMANAGER_H
```

8.13 installationdialog.cpp File Reference

```
#include "installationdialog.h"
#include <QCoreApplication>
#include <QHBoxLayout>
#include <QPushButton>
#include <QSettings>
#include <QTextEdit>
#include <QTimer>
```

Include dependency graph for installationdialog.cpp:

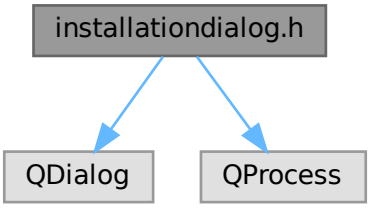


8.14 installationdialog.h File Reference

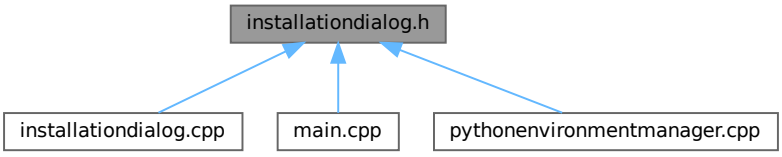
Enthält die Deklaration des InstallationDialog.

```
#include <QDialog>
#include <QProcess>
```

Include dependency graph for installationdialog.h:



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



Classes

- class InstallationDialog

  Ein modaler Dialog, der die Ausgabe des Python-Setup-Skripts anzeigt.

8.14.1 Detailed Description

Enthält die Deklaration des InstallationDialog.

Author

Mike Wild

8.15 installationdialog.h

Go to the documentation of this file.

```
00001
00006 #ifndef INSTALLATIONDIALOG_H
00007 #define INSTALLATIONDIALOG_H
00008
00009 #include <QDialog>
00010 #include <QProcess>
00011
00012 // Forward-Deklarationen
00013 class QTextEdit;
00014 class QPushButton;
00015 class QVBoxLayout;
00016
00024 class InstallationDialog : public QDialog
00025 {
00026     Q_OBJECT
00027 public:
00028     explicit InstallationDialog (QWidget *parent = nullptr);
00029     ~InstallationDialog ();
00030
00031 public slots:
00033     void startPythonSetup ();
00034
00035 signals:
00041     void installationFinished (bool success, const QString &errorMessage = "");
00042
00043 private slots:
00045     void appendOutput ();
00046
00048     void handleProcessFinished (int exitCode, QProcess::ExitStatus exitStatus);
00049
00051     void handleProcessError (QProcess::ProcessError error);
00052
00054     void handleCancelButtonClicked ();
00055
00056 private:
00057     QTextEdit *m_outputDisplay;
00058     QPushButton *m_closeButton;
00059     QProcess *m_setupProcess;
00060 };
00061
00062 #endif // INSTALLATIONDIALOG_H
```

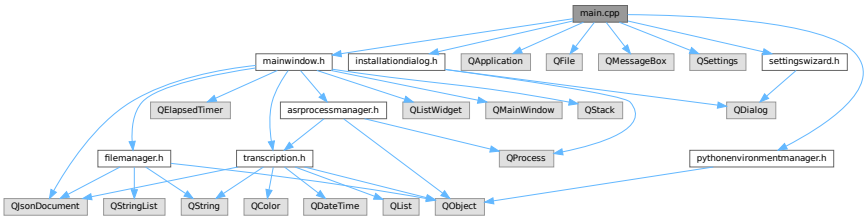
8.16 main.cpp File Reference

Der Haupteinstiegspunkt der Anwendung.

```
#include "installationdialog.h"
#include "mainwindow.h"
```

```
#include <QApplication>
#include <QFile>
#include <QMessageBox>
#include <QSettings>
#include "pythonenvironmentmanager.h"
#include "settingswizard.h"
```

Include dependency graph for main.cpp:



Functions

- int main (int argc, char ∗argv[ ])

8.16.1 Detailed Description

Der Haupteinstiegspunkt der Anwendung.

Author

Mike Wild

8.16.2 Function Documentation

8.16.2.1 main()

```
int main (
            int argc,
            char * argv[] )
```

8.17 mainwindow.cpp File Reference

```
#include "mainwindow.h"
#include <QAction>
#include <QBoxLayout>
#include <QCloseEvent>
#include <QDesktopServices>
#include <QDir>
#include <QFile>
#include <QFileDialog>
#include <QInputDialog>
```

```
#include <QLabel>
#include <QLineEdit>
#include <QMenuBar>
#include <QMessageBox>
#include <QProcess>
#include <QPushButton>
#include <QSplitter>
#include <QTextEdit>
#include <QTimer>
#include <QtConcurrent>
#include "audiofactory.h"
#include "capturethread.h"
#include "pythonenvironmentmanager.h"
#include "settingswizard.h"
#include "speakereditordialog.h"
#include "taggeneratormanager.h"
#include "texteditordialog.h"
#include "transcriptpdfexporter.h"
#include "wavwriterthread.h"
```

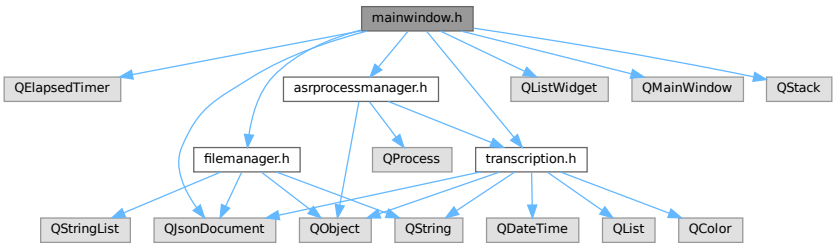
Include dependency graph for mainwindow.cpp:

8.18 mainwindow.h File Reference

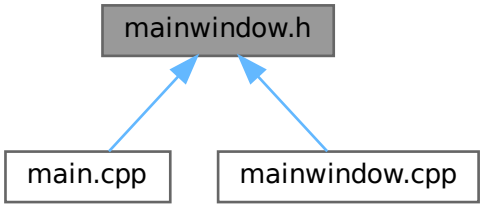
Enthält die Deklaration der MainWindow-Klasse, dem Hauptfenster der Anwendung.

```
#include <QElapsedTimer>
#include <QJsonDocument>
#include <QListWidget>
#include <QMainWindow>
#include <QStack>
#include "asrprocessmanager.h"
#include "filemanager.h"
#include "transcription.h"
```

Include dependency graph for mainwindow.h:



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



Classes

- class MainWindow

  Das Hauptfenster und die zentrale Steuerungseinheit der Anwendung.

8.18.1 Detailed Description

Enthält die Deklaration der MainWindow-Klasse, dem Hauptfenster der Anwendung.

Author

Mike Wild

8.19 mainwindow.h

Go to the documentation of this file.

```
00001
00006 #ifndef MAINWINDOW_H
00007 #define MAINWINDOW_H
00008
00009 #include <QElapsedTimer>
00010 #include <QJsonDocument>
00011 #include <QListWidget>
00012 #include <QMainWindow>
00013 #include <QStack>
00014
00015 // Eigene Klassen
00016 #include "asrprocessmanager.h"
00017 #include "filemanager.h"
00018 #include "transcription.h"
00019
00020 // Forward-Deklarationen
00021 class TagGeneratorManager;
00022 class SpeakerEditorDialog;
00023 class TextEditorDialog;
00024 class WavWriterThread;
00025 class CaptureThread;
00026 class SettingsWizard;
00027 class QAction;
00028 class QCloseEvent;
00029 class QHBoxLayout;
00030 class QLabel;
00031 class QLineEdit;
00032 class QMenuBar;
00033 class QProcess;
```

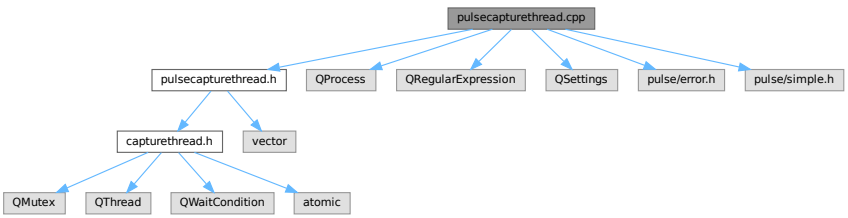
```
00034 class QPushButton;
00035 class QSplitter;
00036 class QTextEdit;
00037 class QTimer;
00038 class QVBoxLayout;
00039
00048 class MainWindow : public QMainWindow
00049 {
00050     Q_OBJECT
00051 public:
00052     explicit MainWindow (QWidget *parent = nullptr);
00053     ~MainWindow () override;
00054
00055 protected:
00062     void closeEvent (QCloseEvent *event) override;
00063
00064 private slots:
00066     void onStartClicked ();
00067
00069     void onStopClicked ();
00070
00076     void onPollTranscripts ();
00077
00079     void onSaveAudio ();
00080
00082     void onSavePDF ();
00083
00085     void onEditSpeakers ();
00086
00088     void onGenerateTags ();
00089
00096     void onMeetingSelected (QListWidgetItem *item);
00097
00099     void onSearchTextChanged (const QString &text);
00100
00102     void processAudio ();
00103
00105     void openSettingsWizard ();
00106
00108     void setStatus (const QString &text, bool keep = false);
00109
00111     void onEditTranscript ();
00112
00114     void onUndo ();
00115
00117     void onRedo ();
00118
00124     void loadTranscriptionFromJson ();
00125
00131     void saveTranscriptionToJson ();
00132
00138     void saveTranscriptionToJsonAs ();
00139
00145     void restoreOriginalTranscription ();
00146
00148     void updateUndoRedoState ();
00149
00151     void setMeetingName (const QString &name);
00152
00154     void onSetMeetingName ();
00155
00160     void onReinstallPython ();
00161
00162 private:
00164     void setupUI ();
00165
00167     void doConnects ();
00168
00174     void loadMeetings ();
00175
00177     void updateUiForCurrentMeeting ();
00178
00180     void filterMeetings (const QString &filter);
00181
00183     QString currentName () const;
00184
00185     // Undo/Redo-Logik
00186     QStack<QJsonDocument> m_undoStack;
00187     QStack<QJsonDocument> m_redoStack;
00188     QAction *m_undoAction;
00189     QAction *m_redoAction;
00190
00191     //  Menü-Aktionen
00192     QAction *m_actionOpen;
00193     QAction *m_actionSave;
00194     QAction *m_actionSaveAs;
00195     QAction *m_actionRestoreOriginal;
```

```
00196     QAction *m_actionClose;
00197     QAction *m_actionSetMeetingName;
00198     QAction *m_settingsAction;
00199     QAction *m_reinstallPythonAction;
00200
00201     // Threads und Manager
00202     CaptureThread *m_captureThread;
00203     WavWriterThread *m_wavWriter;
00204     Transcription *m_script;
00205     FileManager *m_fileManager;
00206     AsrProcessManager *m_asrManager;
00207     TagGeneratorManager *m_tagGenerator;
00208
00209     // UI-Widgets
00210     QSplitter *splitter;
00211     QWidget *leftPanel;
00212     QLineEdit *searchBox;
00213     QListWidget *meetingList;
00214     QWidget *rightPanel;
00215     QVBoxLayout *mainLayout;
00216     QHBoxLayout *buttonLayout;
00217     QPushButton *startButton;
00218     QPushButton *stopButton;
00219     QPushButton *saveAudioButton;
00220     QPushButton *savePDFButton;
00221     QPushButton *assignNamesButton;
00222     QPushButton *generateTagsButton;
00223     QPushButton *editTextButton;
00224     QLabel *timeLabel;
00225     QLabel *nameLabel;
00226     QLabel *statusLabel;
00227     QTextEdit *transcriptView;
00228     QTimer *pollTimer;
00229     QTimer *timeUpdateTimer;
00230     QTimer *statusTimer;
00231     QElapsedTimer elapsedTime;
00232     SpeakerEditorDialog *m_speakerEditorDialog;
00233     TextEditorDialog *m_textEditorDialog;
00234
00235     // Zustandsvariablen
00236     QString m_currentAudioPath;
00237     QString m_currentMeetingName;
00238     QString m_currentMeetingDateTime;
00239
00240     QProcess *pluginProcess;
00241 };
00242
00243 #endif // MAINWINDOW_H
```

8.20 pulsecapturethread.cpp File Reference

```
#include "pulsecapturethread.h"
#include <QProcess>
#include <QRegularExpression>
#include <QSettings>
#include <pulse/error.h>
#include <pulse/simple.h>
```

Include dependency graph for pulsecapturethread.cpp:

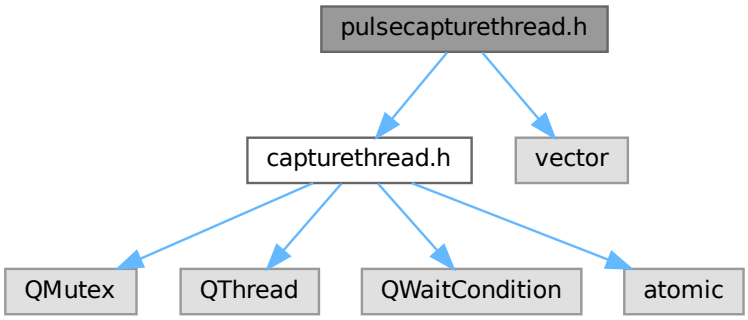


8.21 pulsecapturethread.h File Reference

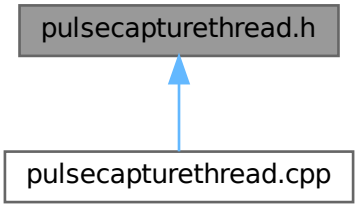
Enthält die Deklaration der PulseCaptureThread-Klasse für die Audioaufnahme unter Linux.

```
#include "capturethread.h"
#include <vector>
```

Include dependency graph for pulsecapturethread.h:



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



Classes

- class PulseCaptureThread

  Eine konkrete Implementierung von CaptureThread für Linux-Systeme mit PulseAudio.

8.21.1 Detailed Description

Enthält die Deklaration der PulseCaptureThread-Klasse für die Audioaufnahme unter Linux.

Author

Mike Wild

8.22 pulsecapturethread.h

Go to the documentation of this file.

```
00001
00006 #ifndef PULSECAPTURETHREAD_H
00007 #define PULSECAPTURETHREAD_H
00008
00009 #include "capturethread.h"
00010 #include <vector>
00011
00012 // Forward-Deklaration für die undurchsichtige PulseAudio-Struktur
00013 struct pa_simple;
00014
00025 class PulseCaptureThread : public CaptureThread
00026 {
00027     Q_OBJECT
00028 public:
00033     explicit PulseCaptureThread (QObject *parent = nullptr);
00034
00035 protected:
00043     bool initializeCapture () override;
00044
00052     void captureLoopIteration () override;
00053
00060     void cleanupCapture () override;
00061
00062 private:
00063     pa_simple *m_paSys;
00064     pa_simple *m_paMic;
00065     int m_modNull;
00066     int m_modLoop;
00067
00068     std::vector<float> bufSys, bufMic, bufMix;
00069     float m_sysGain, m_micGain;
00070 };
00071
00072 #endif // PULSECAPTURETHREAD_H
```

8.23 python/demo.py File Reference

Namespaces

- namespace demo

Variables

- str demo.text

8.24 python/generate_tags.py File Reference

Namespaces

- namespace generate_tags

Functions

- generate_tags.generate_tags (text)

Variables

- generate_tags.input_text = sys.stdin.read()
- generate_tags.generated_tags = generate_tags(input_text)

8.25 python/run_asr.py File Reference

Namespaces

- namespace run_asr

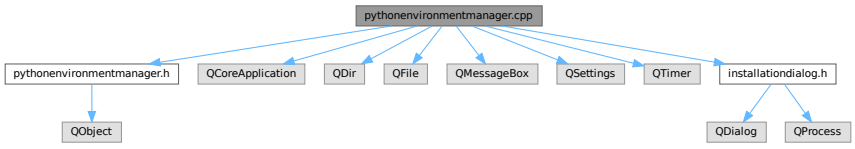
Functions

- run_asr.assign_speakers (transcript_segments, diarization)
- run_asr.main ()

8.26 pythonenvironmentmanager.cpp File Reference

```
#include "pythonenvironmentmanager.h"
#include <QCoreApplication>
#include <QDir>
#include <QFile>
#include <QMessageBox>
#include <QSettings>
#include <QTimer>
#include "installationdialog.h"
```

Include dependency graph for pythonenvironmentmanager.cpp:

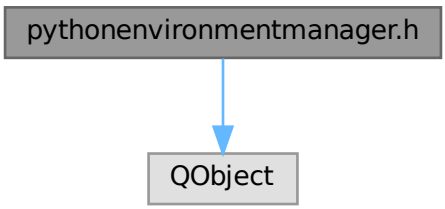


8.27 pythonenvironmentmanager.h File Reference

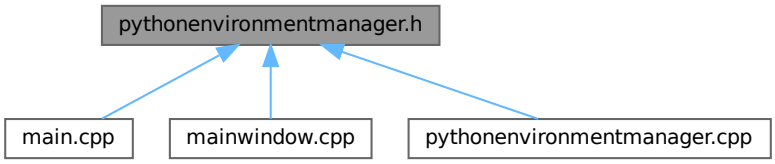
Enthält die Deklaration der PythonEnvironmentManager-Klasse.

```
#include <QObject>
```

Include dependency graph for pythonenvironmentmanager.h:



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



Classes

- class PythonEnvironmentManager

  Verwaltet die Python-Umgebung und deren Installation/Prüfung.

8.27.1 Detailed Description

Enthält die Deklaration der PythonEnvironmentManager-Klasse.

Author

Mike Wild

8.28 pythonenvironmentmanager.h

Go to the documentation of this file.

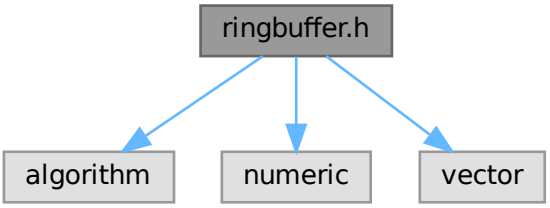
```
00001
00006 #ifndef PYTHONENVIRONMENTMANAGER_H
00007 #define PYTHONENVIRONMENTMANAGER_H
00008
00009 #include <QObject>
00010
00011 class QWidget; // Forward-Deklaration
00012
00022 class PythonEnvironmentManager : public QObject
00023 {
00024     Q_OBJECT
00025 public:
00026     explicit PythonEnvironmentManager (QObject *parent = nullptr);
00027
00042     bool checkAndSetup (bool forceReinstall = false, QWidget *parentWidget = nullptr);
00043
00044 private slots:
00053     void handleInstallationDialogFinished (bool success, const QString &errorMessage);
00054
00055 private:
00061     bool removeVirtualEnvironment (const QString &venvPath);
00062
00063     bool m_dialogSuccess;
00064     QString m_dialogErrorMessage;
00065 };
00066
00067 #endif // PYTHONENVIRONMENTMANAGER_H
```

8.29 ringbuffer.h File Reference

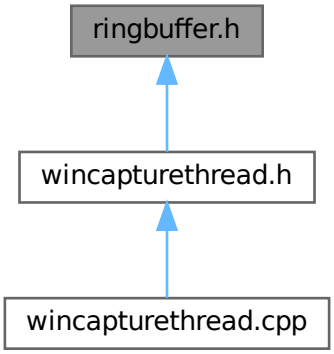
Enthält die Deklaration und Implementierung einer Ringpuffer-Klasse.

```
#include <algorithm>
#include <numeric>
#include <vector>
```

Include dependency graph for ringbuffer.h:



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



Classes

- class RingBuffer

  Eine einfache und effiziente Ringpuffer-Implementierung für float-Werte.

8.29.1 Detailed Description

Enthält die Deklaration und Implementierung einer Ringpuffer-Klasse.

Author

Mike Wild

8.30 ringbuffer.h

Go to the documentation of this file.

```
00001
00006 #ifndef RINGBUFFER_H
00007 #define RINGBUFFER_H
00008
00009 #include <algorithm>
00010 #include <numeric>
00011 #include <vector>
00012
00022 class RingBuffer
00023 {
00024 public:
00029     explicit RingBuffer (
00030         size_t capacity = 0)
00031     {
00032         m_buffer.resize (capacity);
00033     }
00034
00039     void resize (
00040         size_t capacity)
00041     {
00042         m_buffer.resize (capacity);
00043         clear ();
00044     }
00045
00049     void clear ()
00050     {
00051         m_head = 0;
00052         m_tail = 0;
00053         m_size = 0;
00054     }
00055
00060     size_t size () const { return m_size; }
00061
00066     size_t capacity () const { return m_buffer.size (); }
00067
00076     void write (
00077         const float* data, size_t count)
00078     {
00079         if (count == 0 || capacity () == 0)
00080         {
00081             return;
00082         }
00083
00084         // Wenn wir mehr Daten schreiben, als freier Platz vorhanden ist,
00085         // müssen wir die ältesten Daten "vergessen", indem wir den Lesezeiger (tail) nach vorne
      schieben.
00086         size_t free_space = capacity () - m_size;
00087         if (count > free_space)
00088         {
00089             size_t overwrite_count = count - free_space;
00090             // Verschiebe den Lesezeiger um die Anzahl der zu überschreibenden Elemente.
00091             // Der Modulo-Operator sorgt für das "Herumwickeln" am Ende des Puffers.
00092             m_tail = (m_tail + overwrite_count) % capacity ();
00093         }
00094
00095         // Schreibe die neuen Daten an der aktuellen Schreibposition (head).
00096         for (size_t i = 0; i < count; ++i)
00097         {
00098             m_buffer[m_head] = data[i];
00099             // Verschiebe den Schreibzeiger und wickle ihn bei Bedarf um.
00100             m_head = (m_head + 1) % capacity ();
00101         }
00102
00103         // Aktualisiere die Größe des Puffers. Sie kann nicht größer als die Kapazität werden.
00104         m_size = std::min (m_size + count, capacity ());
00105     }
00106
00116     float sampleAt (
00117         double pos) const
00118     {
00119         // Für eine Interpolation benötigen wir mindestens zwei Punkte.
00120         if (m_size < 2)
00121         {
00122             return (m_size == 1) ? m_buffer[m_tail] : 0.0f;
00123         }
00124
00125         // Finde die beiden umgebenden ganzzahligen Indizes.
00126         size_t index0 = static_cast<size_t> (pos);
00127         size_t index1 = index0 + 1;
00128
00129         // Sicherstellen, dass wir nicht über das Ende der gültigen Daten hinaus lesen.
00130         if (index1 >= m_size)
```

```
00131         {
00132             // Am Rand geben wir einfach den letzten gültigen Wert zurück, um Klicks zu vermeiden.
00133             return m_buffer[(m_tail + m_size - 1) % capacity ()];
00134         }
00135 
00136         // Berechne den Anteil zwischen den beiden Punkten (z.B. 0.75 für eine Position von 4.75).
00137         float frac = static_cast<float> (pos - index0);
00138 
00139         // Hole die beiden Sample-Werte. Wichtig: Die Indizes müssen relativ
00140         // zum aktuellen Lesezeiger (m_tail) berechnet werden.
00141         float s0 = m_buffer[(m_tail + index0) % capacity ()];
00142         float s1 = m_buffer[(m_tail + index1) % capacity ()];
00143 
00144         // Lineare Interpolation: Starte bei s0 und addiere den Bruchteil der Differenz zu s1.
00145         return s0 + frac * (s1 - s0);
00146     }
00147 
00154     void consume (
00155         size_t count)
00156     {
00157         // Wir können nicht mehr Elemente entfernen, als vorhanden sind.
00158         count = std::min (count, m_size);
00159         // Verschiebe den Lesezeiger und wickle ihn bei Bedarf um.
00160         m_tail = (m_tail + count) % capacity ();
00161         m_size -= count;
00162     }
00163 
00164 private:
00165     std::vector<float> m_buffer;
00166     size_t m_head = 0;
00167     size_t m_tail = 0;
00168     size_t m_size = 0;
00169 };
00170 
00171 #endif // RINGBUFFER_H
```

8.31 settingswizard.cpp File Reference

```
#include "settingswizard.h"
#include <QDoubleSpinBox>
#include <QFileDialog>
#include <QFontComboBox>
#include <QFormLayout>
#include <QGroupBox>
#include <QLabel>
#include <QLineEdit>
#include <QPushButton>
#include <QSettings>
#include <QSlider>
#include <QVBoxLayout>
#include "filemanager.h"
#include <cmath>
```

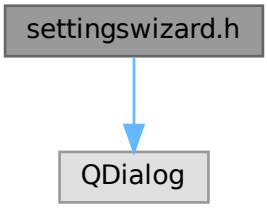
Include dependency graph for settingswizard.cpp:

8.32 settingswizard.h File Reference

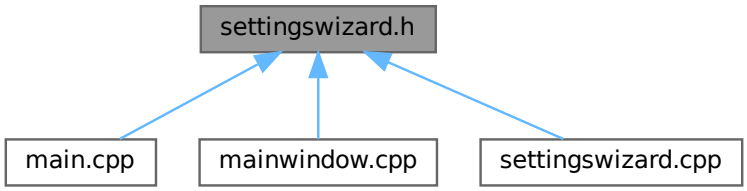
Enthält die Deklaration des SettingsWizard-Dialogs zur Konfiguration der Anwendung.

```
#include <QDialog>
```

Include dependency graph for settingswizard.h:



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



Classes

- class SettingsWizard

  Ein Dialogfenster zur Bearbeitung der Anwendungseinstellungen.

8.32.1 Detailed Description

Enthält die Deklaration des SettingsWizard-Dialogs zur Konfiguration der Anwendung.

Author

Mike Wild

8.33 settingswizard.h

Go to the documentation of this file.

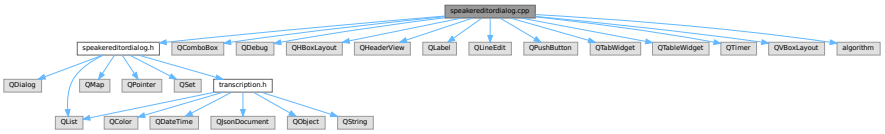
```
00001
00006 #ifndef SETTINGSWIZARD_H
00007 #define SETTINGSWIZARD_H
00008
00009 #include <QDialog>
00010
00011 // Forward-Deklarationen für alle verwendeten Widget-Typen
00012 class QLineEdit;
00013 class QSlider;
00014 class QLabel;
00015 class QDoubleSpinBox;
00016 class QSpinBox;
00017 class QFontComboBox;
00018
00026 class SettingsWizard : public QDialog
00027 {
00028     Q_OBJECT
00029 public:
00034     explicit SettingsWizard (QWidget *parent = nullptr);
00035
00036 private slots:
00038     void saveSettings ();
00039
00041     void updateDurationLabel (int value);
00042
00044     void syncSysGainSlider (double value);
00045
00047     void syncMicGainSlider (double value);
00048
00050     void syncSysGainSpin (int sliderValue);
00051
00053     void syncMicGainSpin (int sliderValue);
00054
00055 private:
00061     int validateBufferSize (int kb);
00062
00063     // --- UI-Elemente ---
00064     // Pfad- und Buffer-Einstellungen
00065     QLineEdit *pythonEdit;
00066     QLineEdit *scriptEdit;
00067     QLineEdit *wavEdit;
00068     QLineEdit *asrWavEdit;
00069     QLineEdit *meetingEdit;
00070     QSlider *bufferSlider;
00071     QLabel *durationLabel;
00072
00073     // Audio-Verstärkung
00074     QDoubleSpinBox *sysGainSpin;
00075     QSlider *sysGainSlider;
00076     QDoubleSpinBox *micGainSpin;
00077     QSlider *micGainSlider;
00078
00079     // PDF-Exporteinstellungen
00080     QSpinBox *pdfHeadlineSpin;
00081     QSpinBox *pdfBodySpin;
00082     QSpinBox *pdfMetaSpin;
00083     QSpinBox *marginTopSpin;
00084     QSpinBox *marginRightSpin;
00085     QSpinBox *marginBottomSpin;
00086     QSpinBox *marginLeftSpin;
00087     QFontComboBox *fontFamilyCombo;
00088 };
00089
00090 #endif // SETTINGSWIZARD_H
```

8.34 speakereditordialog.cpp File Reference

```
#include "speakereditordialog.h"
#include <QComboBox>
#include <QDebug>
#include <QHBoxLayout>
#include <QHeaderView>
#include <QLabel>
```

```
#include <QLineEdit>
#include <QPushButton>
#include <QTabWidget>
#include <QTableWidget>
#include <QTimer>
#include <QVBoxLayout>
#include <algorithm>
```

Include dependency graph for speakereditordialog.cpp:

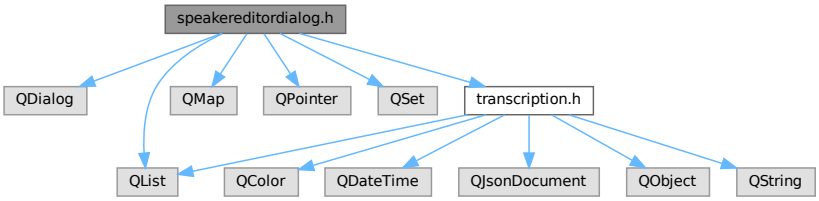


8.35 speakereditordialog.h File Reference

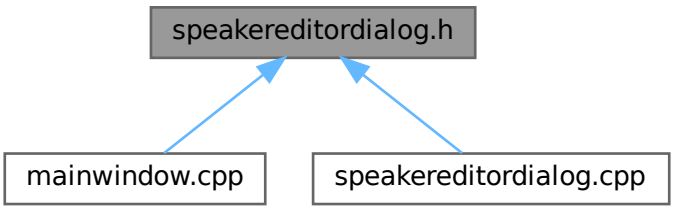
Enthält die Deklaration des SpeakerEditorDialog zur Bearbeitung von Sprechernamen.

```
#include <QDialog>
#include <QList>
#include <QMap>
#include <QPointer>
#include <QSet>
#include "transcription.h"
```

Include dependency graph for speakereditordialog.h:



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



Classes

- class SpeakerEditorDialog

  Ein nicht-modaler Dialog zur Bearbeitung von Sprecherinformationen im Transkript.

8.35.1 Detailed Description

Enthält die Deklaration des SpeakerEditorDialog zur Bearbeitung von Sprechernamen.

Author

Mike Wild

8.36 speakereditordialog.h

Go to the documentation of this file.

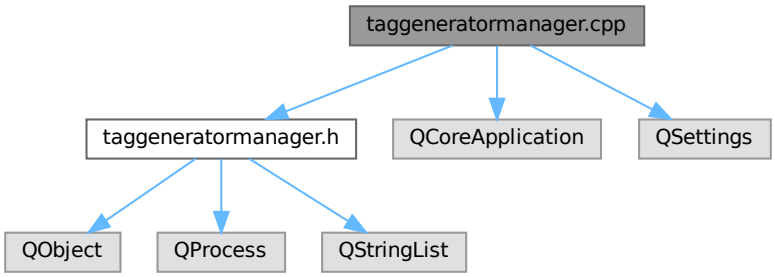
```
00001
00006 #ifndef SPEAKEREDITORDIALOG_H
00007 #define SPEAKEREDITORDIALOG_H
00008
00009 #include <QDialog>
00010 #include <QList>
00011 #include <QMap>
00012 #include <QPointer>
00013 #include <QSet>
00014
00015 #include "transcription.h"
00016
00017 // Forward-Deklarationen für UI-Elemente
00018 class QTableWidgetItem;
00019 class QTableWidget;
00020 class QComboBox;
00021 class QPushButton;
00022 class QLabel;
00023 class QTabWidget;
00024 class QLineEdit;
00025 class QTimer;
00026
00036 class SpeakerEditorDialog : public QDialog
00037 {
00038     Q_OBJECT
00039 public:
00045     explicit SpeakerEditorDialog (Transcription* transcription, QWidget* parent = nullptr);
00046
00052     void setSelectedSegment (const QString& start, const QString& end);
00053
00054 public slots:
00060     void onTranscriptionChanged ();
00061
00062 private slots:
00064     void setupUI ();
00065
00067     void populateGlobalSpeakerTable ();
00068
00070     void populateSegmentTable ();
00071
00073     void handleApplyOkButtonClicked ();
00074
00076     void handleCancelButtonClicked ();
00077
00079     void setDialogStatus (const QString& text, bool temporary = true);
00080
00082     void onSegmentSpeakerChanged (int index);
00083
00085     void onGlobalSpeakerNameChanged (const QString& text);
00086
00088     void onMergeSpeakersClicked ();
00089
00090 private:
00092     void applyCurrentTabChanges ();
00093
00095     void updateKnownSpeakers ();
```

```

    // --- Member-Variablen ---
    QPointer<Transcription> m_transcription;

    // UI-Elemente
    QTabWidget* m_tabWidget;
    QTableWidget* m_globalSpeakerTable;
    QTableWidget* m_segmentTable;
    QPushButton* m_applyButton;
    QPushButton* m_okButton;
    QPushButton* m_cancelButton;
    QLabel* m_statusLabel;
    QTimer* m_statusTimer;
    QLineEdit* m_mergeNameEdit;
    QPushButton* m_mergeSpeakersButton;

    // Interne Zustands- und Puffer-Variablen
    QSet<QString> m_allKnownSpeakers;
    QMap<QString, QString> m_currentGlobalNames;
    QMap<QPair<QString, QString>, QString>
        m_currentSegmentNames;

    QString m_selectedSegmentStart;
    QString m_selectedSegmentEnd;
};

#endif // SPEAKEREDITORDIALOG_H
```

8.37 taggeneratormanager.cpp File Reference

```
#include "taggeneratormanager.h"
#include <QCoreApplication>
#include <QSettings>
```

Include dependency graph for taggeneratormanager.cpp:



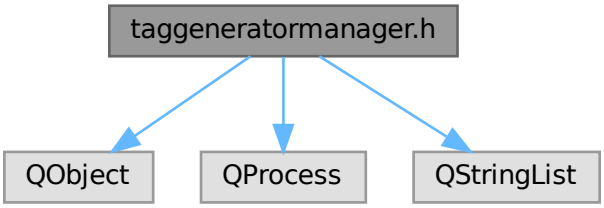
8.38 taggeneratormanager.h File Reference

Enthält die Deklaration der TagGeneratorManager-Klasse.

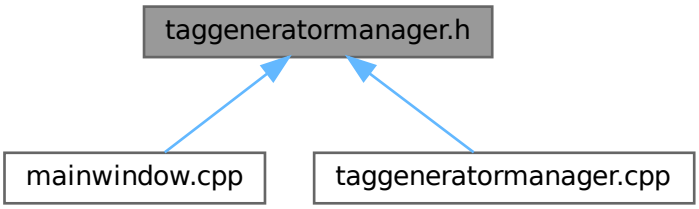
```
#include <QObject>
#include <QProcess>
```

```
#include <QStringList>
```

Include dependency graph for taggeneratormanager.h:



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



Classes

- class TagGeneratorManager

  Steuert den externen Python-Prozess zur automatischen Tag-Erstellung.

8.38.1 Detailed Description

Enthält die Deklaration der TagGeneratorManager-Klasse.

Author

Mike Wild

8.39 taggeneratormanager.h

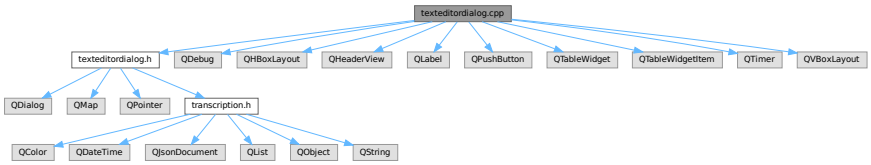
Go to the documentation of this file.

```
00001
00006 #ifndef TAGGENERATORMANAGER_H
00007 #define TAGGENERATORMANAGER_H
00008
00009 #include <QObject>
00010 #include <QProcess>
00011 #include <QStringList>
00012
00021 class TagGeneratorManager : public QObject
00022 {
00023     Q_OBJECT
00024 public:
00029     explicit TagGeneratorManager (QObject *parent = nullptr);
00030
00031 public slots:
00039     void generateTagsFor (const QString &fullText);
00040
00041 signals:
00048     void tagsReady (const QStringList &tags, bool success, const QString &errorMsg = "");
00049
00050 private slots:
00057     void onProcessFinished (int exitCode, QProcess::ExitStatus exitStatus);
00058
00059 private:
00060     QProcess *m_process;
00061     QString m_pythonPath;
00062     QString m_scriptPath;
00063 };
00064
00065 #endif // TAGGENERATORMANAGER_H
```

8.40 texteditordialog.cpp File Reference

```
#include "texteditordialog.h"
#include <QDebug>
#include <QHBoxLayout>
#include <QHeaderView>
#include <QLabel>
#include <QPushButton>
#include <QTableWidget>
#include <QTableWidgetItem>
#include <QTimer>
#include <QVBoxLayout>
```

Include dependency graph for texteditordialog.cpp:



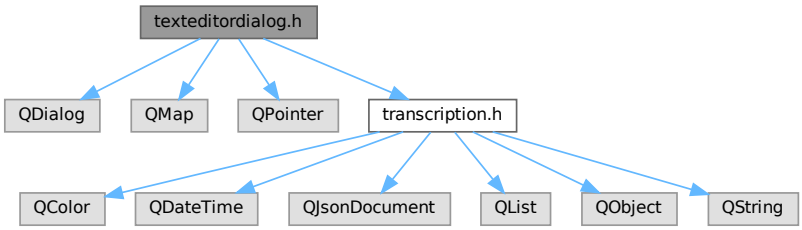
8.41 texteditordialog.h File Reference

Enthält die Deklaration des TextEditorDialog zur Bearbeitung des Transkript-Textes.

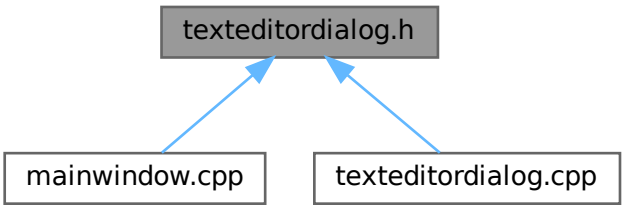
```
#include <QDialog>
#include <QMap>
```

```
#include <QPointer>
#include "transcription.h"
```

Include dependency graph for texteditordialog.h:



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



Classes

- class TextEditorDialog

  Ein nicht-modaler Dialog zur direkten Bearbeitung der Textinhalte von Transkript-Segmenten.

8.41.1 Detailed Description

Enthält die Deklaration des TextEditorDialog zur Bearbeitung des Transkript-Textes.

Author

Mike Wild

8.42 texteditordialog.h

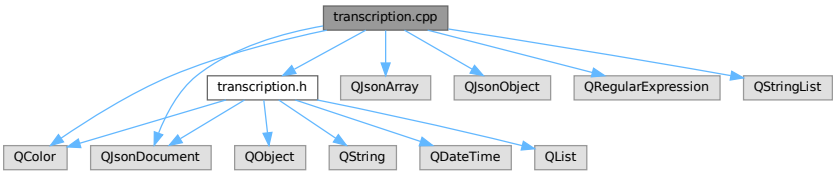
Go to the documentation of this file.

```
00001
00006 #ifndef TEXTEDITORDIALOG_H
00007 #define TEXTEDITORDIALOG_H
00008
00009 #include <QDialog>
00010 #include <QMap>
00011 #include <QPointer>
00012
00013 #include "transcription.h"
00014
00015 // Forward-Deklarationen
00016 class QTableWidget;
00017 class QPushButton;
00018 class QLabel;
00019 class QTimer;
00020 class QTableWidgetItem;
00021 class QLineEdit;
00022
00030 class TextEditorDialog : public QDialog
00031 {
00032     Q_OBJECT
00033 public:
00039     explicit TextEditorDialog (Transcription* transcription, QWidget* parent = nullptr);
00040
00041 public slots:
00046     void onTranscriptionChanged ();
00047
00048 private slots:
00050     void applyChanges ();
00051
00053     void handleApplyButtonClicked ();
00054
00056     void handleOkButtonClicked ();
00057
00059     void handleCancelButtonClicked ();
00060
00062     void onTextItemChanged (QTableWidgetItem* item);
00063
00064 private:
00066     void setupUI ();
00067
00069     void populateTable ();
00070
00072     void setDialogStatus (const QString& text, bool temporary = true);
00073
00074     // --- Member-Variablen ---
00075     QPointer<Transcription> m_transcription;
00076
00077     // UI-Elemente
00078     QTableWidget* m_table;
00079     QPushButton* m_applyButton;
00080     QPushButton* m_okButton;
00081     QPushButton* m_cancelButton;
00082     QLabel* m_statusLabel;
00083     QTimer* m_statusTimer;
00084
00089     QMap<QPair<QString, QString>, QString> m_pendingTextChanges;
00090 };
00091
00092 #endif // TEXTEDITORDIALOG_H
```

8.43 transcription.cpp File Reference

```
#include "transcription.h"
#include <QColor>
#include <QJsonArray>
#include <QJsonDocument>
#include <QJsonObject>
#include <QRegularExpression>
```

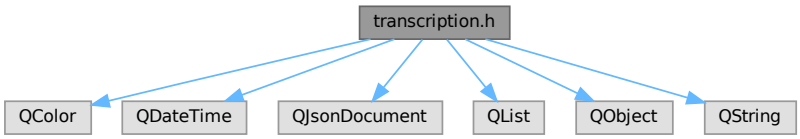
```
#include <QStringList>
```
Include dependency graph for transcription.cpp:



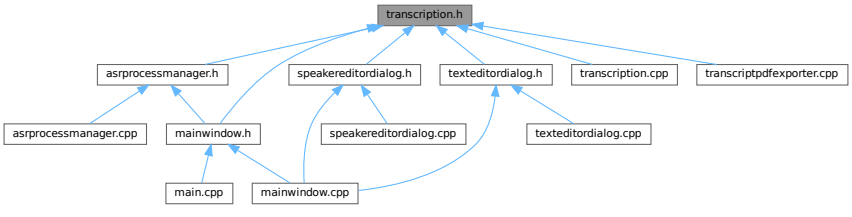
8.44 transcription.h File Reference

Enthält die Deklaration der Datenmodell-Klassen Transcription und MetaText.

```
#include <QColor>
#include <QDateTime>
#include <QJsonDocument>
#include <QList>
#include <QObject>
#include <QString>
```
Include dependency graph for transcription.h:



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



Classes

- struct MetaText

  Eine einfache Datenstruktur, die ein einzelnes Segment eines Transkripts repräsentiert.
- class Transcription

  Das zentrale Datenmodell für ein komplettes Meeting-Transkript.

8.44.1 Detailed Description

Enthält die Deklaration der Datenmodell-Klassen Transcription und MetaText.

Author

Mike Wild

8.45 transcription.h

Go to the documentation of this file.

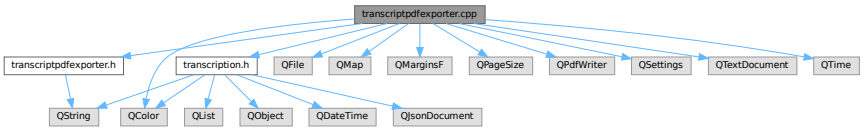
```
00001
00006 #ifndef TRANSCRIPTION_H
00007 #define TRANSCRIPTION_H
00008
00009 #include <QColor>
00010 #include <QDateTime>
00011 #include <QJsonDocument> // Nötig für den Rückgabetyp von toJson()
00012 #include <QList>
00013 #include <QObject>
00014 #include <QString>
00015
00022 struct MetaText
00023 {
00024     MetaText () = default;
00025     MetaText (
00026         const QString &start, const QString &end, const QString &speaker, const QString &text)
00027         : Speaker (speaker)
00028         , Text (text)
00029         , Start (start)
00030         , End (end)
00031     {
00032     }
00033
00034     QString Speaker;
00035     QString Text;
00036     QString Start;
00037     QString End;
00038     QStringList Tags;
00039
00040     void addTag (
00041         const QString &tag)
00042     {
00043         if (!Tags.contains (tag))
00044             Tags.append (tag);
00045     }
00046     void removeTag (
00047         const QString &tag)
00048     {
00049         Tags.removeAll (tag);
00050     }
00051     bool hasTag (
00052         const QString &tag) const
00053     {
00054         return Tags.contains (tag);
00055     }
00056 };
00057
00068 class Transcription : public QObject
00069 {
00070     Q_OBJECT
00071 public:
00072     explicit Transcription (QObject *parent = nullptr);
00073
00075     QString text () const;
00076
00083     QString script () const;
00084
00086     bool changeSpeaker (const QString &oldSpeaker, const QString &newSpeaker);
00087
00089     bool changeText (const QString &start, const QString &end, const QString &newText);
00090
00092     bool changeSpeakerForSegment (const QString &start,
00093                                   const QString &end,
00094                                   const QString &newSpeaker);
00095
00097     const QList<MetaText> &getMetaTexts () const { return m_content; }
```

```
00098
00100     QJsonDocument toJson () const;
00101
00103     bool fromJson (const QByteArray &data);
00104
00105     // --- Getter für Metadaten ---
00106     QString name () const { return m_meetingName; }
00107     QDateTime dateTime () const { return m_startTime; }
00108     QString getDurationAsString () const;
00109
00110     // --- Tag-Management ---
00111     QStringList tags () const { return m_tags; }
00112     void setTags (const QStringList &tags);
00113     void addTag (const QString &tag);
00114     void removeTag (const QString &tag);
00115     bool hasTag (const QString &tag) const;
00116     QList<MetaText> segmentsWithTag (const QString &tag) const;
00117
00118 public slots:
00120     void add (const MetaText &part);
00121
00123     void clear ();
00124
00126     void beginBatchUpdate ();
00127
00129     void endBatchUpdate ();
00130
00132     void setName (const QString &name);
00133
00135     void setDateTime (QDateTime dateTime);
00136
00137 signals:
00139     void changed ();
00140
00142     void edited ();
00143
00144 private:
00146     QColor speakerColor (const QString &speaker) const;
00147
00148     QList<MetaText> m_content;
00149     QStringList m_tags;
00150
00151     // Zähler für den internen Zustand
00152     int m_unknownCounter{0};
00153     int m_batchUpdateCounter{0};
00154     bool m_changesPending
00155         = false;
00156
00157     // Meeting-Metadaten
00158     QString m_meetingName;
00159     QDateTime m_startTime;
00160 };
00161
00162 #endif // TRANSCRIPTION_H
```

8.46 transcriptpdfexporter.cpp File Reference

```
#include "transcriptpdfexporter.h"
#include "transcription.h"
#include <QColor>
#include <QFile>
#include <QMap>
#include <QMarginsF>
#include <QPageSize>
#include <QPdfWriter>
#include <QSettings>
#include <QTextDocument>
#include <QTime>
```

Include dependency graph for transcriptpdfexporter.cpp:

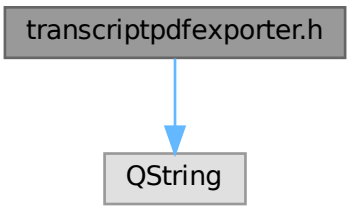


8.47 transcriptpdfexporter.h File Reference

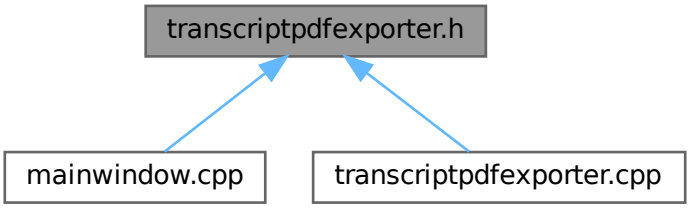
Enthält die Deklaration der TranscriptPdfExporter-Klasse.

```
#include <QString>
```

Include dependency graph for transcriptpdfexporter.h:



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



Classes

- class TranscriptPdfExporter

  Erstellt eine formatierte, mehrseitige PDF-Repräsentation eines Transcription-Objekts.

8.47.1 Detailed Description

Enthält die Deklaration der TranscriptPdfExporter-Klasse.

Author

Mike Wild

8.48 transcriptpdfexporter.h

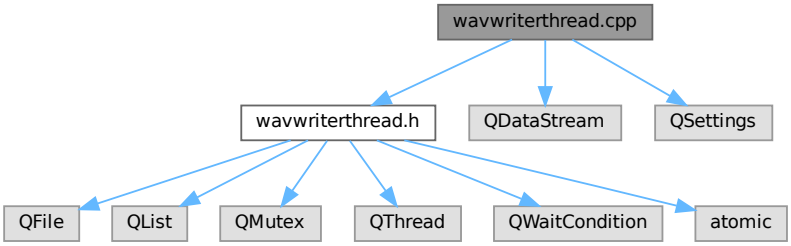
Go to the documentation of this file.

```
00001
00006 #ifndef TRANSCRIPTPDFEXPORTER_H
00007 #define TRANSCRIPTPDFEXPORTER_H
00008
00009 #include <QString>
00010
00011 // Forward-Deklarationen
00012 class Transcription;
00013 class QPdfWriter;
00014
00022 class TranscriptPdfExporter
00023 {
00024 public:
00029     explicit TranscriptPdfExporter (const Transcription &transcription);
00030
00040     bool exportToPdf (const QString &filePath) const;
00041
00042 private:
00044     void setupPdfWriter (QPdfWriter &writer) const;
00045
00047     void buildHtmlContent (QString &html) const;
00048
00049     // calculateDuration() ist eine private Hilfsfunktion und muss nicht im Header deklariert werden.
00050
00051     // --- Member-Variablen ---
00052     const Transcription
00053         &m_transcription;
00054
00055     // Geladene Einstellungen für das Layout
00056     int m_fontSizeHeadline;
00057     int m_fontSizeMetadata;
00058     int m_fontSizeBody;
00059     QString m_fontFamily;
00060     int m_marginLeft;
00061     int m_marginTop;
00062     int m_marginRight;
00063     int m_marginBottom;
00064 };
00065
00066 #endif // TRANSCRIPTPDFEXPORTER_H
```

8.49 wavwriterthread.cpp File Reference

```
#include "wavwriterthread.h"
#include <QDataStream>
```

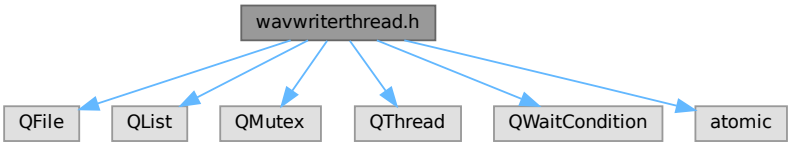
```
#include <QSettings>
```
Include dependency graph for wavwriterthread.cpp:



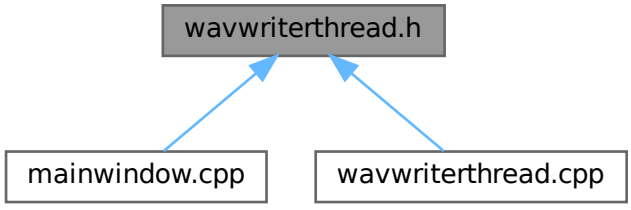
8.50 wavwriterthread.h File Reference

Enthält die Deklaration des WavWriterThread zum Schreiben von Audio-Dateien.

```
#include <QFile>
#include <QList>
#include <QMutex>
#include <QThread>
#include <QWaitCondition>
#include <atomic>
```
Include dependency graph for wavwriterthread.h:



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



Classes

- class WavWriterThread

  Ein dedizierter Thread, der Audio-Daten in WAV-Dateien schreibt.

8.50.1 Detailed Description

Enthält die Deklaration des WavWriterThread zum Schreiben von Audio-Dateien.

Author

Mike Wild

8.51 wavwriterthread.h

Go to the documentation of this file.

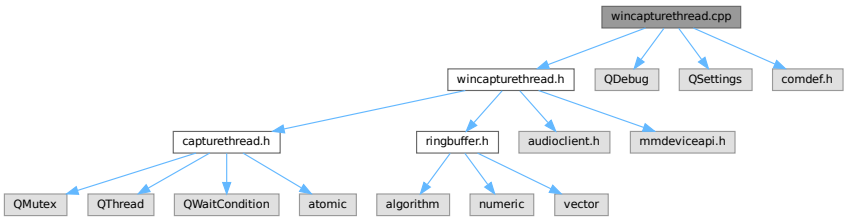
```
00001
00006 #ifndef WAVWRITERTHREAD_H
00007 #define WAVWRITERTHREAD_H
00008
00009 #include <QFile>
00010 #include <QList>
00011 #include <QMutex>
00012 #include <QThread>
00013 #include <QWaitCondition>
00014 #include <atomic>
00015
00026 class WavWriterThread : public QThread
00027 {
00028     Q_OBJECT
00029 public:
00034     explicit WavWriterThread (QObject *parent = nullptr);
00035
00039     ~WavWriterThread ();
00040
00049     void startWriting (const QString &hqPath, const QString &asrPath);
00050
00054     void shutdown ();
00055
00056 public slots:
00063     void writeChunk (QList<float> chunk);
00064
00071     void stopWriting ();
00072
00073 signals:
00078     void finishedWriting ();
00079
00080 protected:
00087     void run () override;
00088
00089 private:
00095     void writeHeaders (qint64 hqBytes, qint64 asrBytes);
00096
00101     void writeCurrentBufferToDisk (QList<float> &buffer);
00102
00103     QFile m_hqFile;
00104     QFile m_asrFile;
00105
00106     // Synchronisation und Zustand
00107     QMutex m_mutex;
00108     QWaitCondition m_mainLoopCond;
00109     QWaitCondition
00110         m_dataAvailableCond;
00111     std::atomic<bool> m_active;
00112     std::atomic<bool> m_shutdown;
00113
00114     // Puffer und Zähler
00115     QList<float> m_bufferFloat;
00116     qint64 m_hqBytesWritten;
00117     qint64 m_asrBytesWritten;
00118     qint64 m_flushThresholdBytes;
00119     int m_downsampleOffset;
00120
```

```
00121     // Audio-Format-Konstanten
00122     const int m_sampleRateHQ;
00123     const int m_channelsHQ;
00124     const int m_bitsPerSampleHQ;
00125     const int m_sampleRateASR;
00126 };
00127
00128 #endif // WAVWRITERTHREAD_H
```

8.52 wincapturethread.cpp File Reference

```
#include "wincapturethread.h"
#include <QDebug>
#include <QSettings>
#include <comdef.h>
```

Include dependency graph for wincapturethread.cpp:

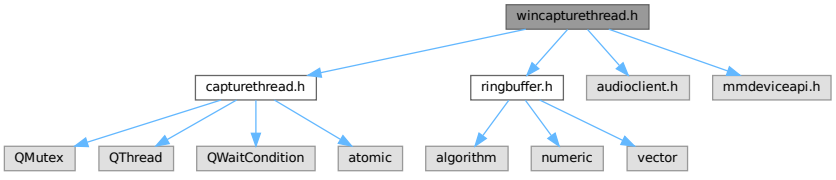


8.53 wincapturethread.h File Reference

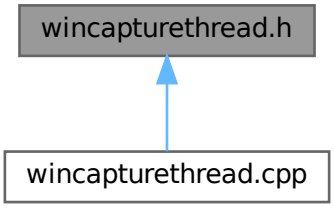
Enthält die Deklaration der WinCaptureThread-Klasse für die Audioaufnahme unter Windows.

```
#include "capturethread.h"
#include "ringbuffer.h"
#include <audioclient.h>
#include <mmdeviceapi.h>
```

Include dependency graph for wincapturethread.h:



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



Classes

- class WinCaptureThread

  Eine konkrete Implementierung von CaptureThread für Windows-Systeme.

8.53.1 Detailed Description

Enthält die Deklaration der WinCaptureThread-Klasse für die Audioaufnahme unter Windows.

Author

Mike Wild

8.54 wincapturethread.h

Go to the documentation of this file.

```
00001
00006 #ifndef WINCAPTURETHREAD_H
00007 #define WINCAPTURETHREAD_H
00008
00009 #include "capturethread.h"
00010 #include "ringbuffer.h"
00011 #include <audioclient.h>
00012 #include <mmdeviceapi.h>
00013
00023 class WinCaptureThread : public CaptureThread
00024 {
00025     Q_OBJECT
00026 public:
00031     explicit WinCaptureThread (QObject *parent = nullptr);
00032
00033 protected:
00039     void run() override;
00040
00045     bool initializeCapture() override;
00046
00053     void captureLoopIteration() override;
00054
00060     void cleanupCapture() override;
00061
00062 private:
00063     // --- COM-Interfaces für System-Audio (Loopback) ---
00064     IAudioClient *m_audioClientSys = nullptr;
00065     IAudioCaptureClient *m_captureClientSys
00066         = nullptr;
```

```
00067     IMMDevice *m_deviceSys = nullptr;
00068     IMMDeviceEnumerator *m_deviceEnumerator
00069         = nullptr;
00070
00071     // --- COM-Interfaces für Mikrofon-Audio ---
00072     IAudioClient *m_audioClientMic = nullptr;
00073     IAudioCaptureClient *m_captureClientMic
00074         = nullptr;
00075     IMMDevice *m_deviceMic = nullptr;
00076
00077     // --- Timing und Resampling ---
00078     LARGE_INTEGER m_perfCounterFreq;
00079     LARGE_INTEGER m_lastTime;
00080     double m_sampleAccumulator
00081         = 0.0;
00082     double m_resampPosMic = 0.0;
00083     double m_resampPosSys = 0.0;
00084     const int m_pollingIntervalMs
00085         = 10;
00086
00087     // --- Ringpuffer ---
00088     RingBuffer m_fifoSysL;
00089     RingBuffer m_fifoMicL;
00090     RingBuffer m_fifoSysR;
00091     RingBuffer m_fifoMicR;
00092
00093     // --- Geräte-Eigenschaften ---
00094     UINT32 m_nativeSampleRateSys = 0;
00095     UINT32 m_nativeChannelsSys = 0;
00096     UINT32 m_nativeSampleRateMic = 0;
00097     UINT32 m_nativeChannelsMic = 0;
00098 };
00099
00100 #endif // WINCAPTURETHREAD_H
```

Index