



VIVOTEK NETWORK DEVELOPMENT PLATFORM

Audio/Video Composer Module

Version 1.0.0.1

2017/01/20

© 2017 VIVOTEK Inc. All Right Reserved

VIVOTEK may make changes to specifications and product descriptions at any time, without notice.

The following is trademarks of VIVOTEK Inc., and may be used to identify VIVOTEK products only: VIVOTEK. Other product and company names contained herein may be trademarks of their respective owners.

No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from VIVOTEK Inc.

TABLE OF CONTENTS

TABLE OF CONTENTS	1
1. OVERVIEW.....	2
1.1 Introduction	2
1.2 Getting Started with AVComposer Module	2
1.3 File Structure	2
2. PROGRAMMER'S GUIDE	3
2.1 Using AVComposer Module	3
3. SAMPLE CODE	4
3.1 Save To 3GP.....	4
4. API REFERENCE	5
4.1 Enumeration.....	5
4.1.1 EAVCOMPOSER_TRACK_TYPE.....	6
4.1.2 EAVCOMPOSER_RECORD_OPTIONS	7
4.2 Data Structure	8
4.2.1 TAVCOMPOSER_TIMEPOINT.....	9
4.2.2 TAVCOMPOSERCloseFileInfo	10
4.3 Callback Function	11
4.3.1 AVCOMPOSER_PF_REQUEST_FILENAME	12
4.3.2 AVCOMPOSER_PF_REQUEST_FILENAMEEW	13
4.4 API Definition	14
4.4.1 AVComposer_CreateHandle	15
4.4.2 AVComposer_CreateHandleW.....	17
4.4.3 AVComposer_ReleaseHandle	19
4.4.4 AVComposer_StartRecording	20
4.4.5 AVComposer_StopRecording.....	22
4.4.6 AVComposer_InputPacket	24
4.4.7 AVComposer_SetAuthenticity	26

1. Overview

1.1 Introduction

This document describes the properties and methods supported by the VIVOTEK Audio Video Composer (AVComposer) module.

1.2 Getting Started with AVComposer Module

The main usage of AVComposer module is to record frames from VIVOTEK camera or video server.

1.3 File Structure

Table 1-1 File Structure

File	Description
doc\VNDP_AVComposer_API.pdf	This manual document
lib\d_AVComposer.lib	The dynamic linking library
lib\AVComposer.dll	The dynamic runtime library
inc\AVComposer.h	Header file
inc\AVComposerDefine.h	Header file
inc\AVComposerErrorDef.h	Header file

2. Programmer's Guide

2.1 Using AVComposer Module

- Starts AVComposer module.
- Records the frames from cameras and video servers.
- Gets new filename while reach the file size or duration condition via a callback function.

3. Sample Code

3.1 Save To 3GP

Description

Save the frames from DataBroker to 3GP files.

Sample Code

- Step 1. Initialize DataBroker.
- Step 2. Create connection.
- Step 3. Define a `GetNewFilename` function to generate specified media file name. Use [AVComposer CreateHandle](#) to create an AVComposer handle and pass `GetNewFilename` as the `AVCOMPOSER_PF_REQUEST_FILENAME` callback function. AVComposer will be ready to save input media packets after calling [AVComposer StartRecording](#).
- Step 4. Connect to a device, DataBroker uses asynchronous mechanism while connecting to device, the connection information would be carried in `DataBrokerStatusCallback` function.
- Step 5. While receiveing media packets in `DataBrokerAVCallback`, it then uses [AVComposer InputPacket](#) to save these media packets to audio or video track.
- Step 6. Disconnect device.
- Step 7. Delete connection.
- Step 8. Release DataBroker.
- Step 9. Use [AVComposer StopRecording](#) to close 3GP file and [AVComposer ReleaseHandle](#) to delete AVComposer handle.

4. API Reference

This chapter describes the API function calls for the AVComposer module.

4.1 Enumeration

The enumeration used is depicted here.

4.1.1 EAVCOMPOSER_TRACK_TYPE

Defines AVComposer media track type.

```
typedef enum {  
  
    eAVComposerTrack1          = 0  
  
    eAVComposerTrack2          = 1  
  
} EAVCOMPOSER_TRACK_TYPE;
```

Values

eAVComposerTrack1

Video track (recommended).

eAVComposerTrack2

Audio track (recommended).

Remarks

By selecting one value of EAVCOMPOSER_TRACK_TYPE, [AVComposer InputPacket](#) will save media packets to the chosen track. The most recommended case is to save video and audio packets to two distinct tracks.

Requirements

AVComposerDefine.h

4.1.2 EAVCOMPOSER_RECORD_OPTIONS

Defines AVComposer record options.

```
typedef enum {  
  
                                eRecOptVivoTrack                = 1  
  
} EAVCOMPOSER_RECORD_OPTIONS;
```

Values

eRecOptVivoTrack

Save the VIVOTEK proprietary data along with media packets. Seted that watermark will take effect.

Remarks

When eRecOptVivoTrack is set, [AVComposer InputPacket](#) will save VIVOTEK proprietary data along with media packets. Applications that need to use information in the proprietary data for special integration should set eRecOptVivoTrack.

Applications that need use AVComposer authentication function should set eRecOptVivoTrack.

Requirements

AVComposerDefine.h

4.2 Data Structure

The data structure is depicted here.

VIVOTEK CONFIDENTIAL
2017.01.20

4.2.1 TAVCOMPOSER_TIMEPOINT

Describes the timestamp.

```
typedef struct {  
  
    DWORD    dwTimeInSecond;  
  
    DWORD    dwMilliSecond;  
  
} TAVCOMPOSER_TIMEPOINT;
```

Members

dwTimeInSecond

The time point, in seconds.

dwMilliSecond

The time point, in milliseconds.

Remarks

Requirements

AVComposerDefine.h

4.2.2 TAVCOMPOSERCloseFileInfo

Describes information of a closed file.

```
typedef struct {
    WORD wSize;
    DWORD dwFileSize;
    TAVCOMPOSER\_TIMEPOINT tStartTime;
    TAVCOMPOSER\_TIMEPOINT tEndTime;
    BOOL bCreatedMediaFile;
} TAVCOMPOSERCloseFileInfo;
```

Members

wSize

The number of bytes required by this structure.

dwFileSize

The size, in bytes of the file.

tStartTime

The start time of the file.

tEndTime

The end time of the file.

bCreatedMediaFile

Indicate if the media file is successfully created.

Remarks

The TAVCOMPOSERCloseFileInfo describes information of a closed file. It will be callback through callback function ([AVCOMPOSER_PF_REQUEST_FILENAME](#), [AVCOMPOSER_PF_REQUEST_FILENAMEW](#)) or be used in [AVComposer_StopRecording](#).

Requirements

AVComposerDefine.h

4.3 Callback Function

The callback function is depicted here.

VIVOTEK CONFIDENTIAL
2017.01.20

4.3.1 AVCOMPOSER_PF_REQUEST_FILENAME

The AVCOMPOSER_PF_REQUEST_FILENAME function is the callback function used within composing media packets to give a new file name. The name AVCOMPOSER_PF_REQUEST_FILENAME is a placeholder for the application-specified function.

Syntax

```
typedef SCODE  
(*AVCOMPOSER_PF_REQUEST_FILENAME)(  
  
                                HANDLE                hContext,  
  
                                char                  *pszNewFilename,  
  
                                TAVCOMPOSERCloseFileInfo *ptCurFileInfo  
  
);
```

Members

hContext

[in] Data to be passed to callback function. This is set through [AVComposer_CreateHandle](#).

***pszNewFilename**

[in] New filename. The maximum length of this string is FILENAME_MAX.

***ptCurFileInfo**

[in] Pointer to the information of latest closed file.

Remarks

Requirements

AVComposerDefine.h

4.3.2 AVCOMPOSER_PF_REQUEST_FILENAMEEW

The AVCOMPOSER_PF_REQUEST_FILENAMEEW function is the callback function used within composing media packets to give a new file name (in Unicode). The name AVCOMPOSER_PF_REQUEST_FILENAMEEW is a placeholder for the application-specified function.

Syntax

```
typedef SCODE
(*AVCOMPOSER_PF_REQUEST_FILENAMEEW)(
    HANDLE                hContext,
    wchar_t               *pszNewFilename,
    TAVCOMPOSERCloseFileInfo *ptCurFileInfo
);
```

Members

hContext

[in] Data to be passed to callback function. This is set through [AVComposer_CreateHandleW](#).

***pszNewFilename**

[in] New filename. The maximum length of this string is FILENAME_MAX.

***ptCurFileInfo**

[in] Pointer to the information of latest closed file.

Remarks

Requirements

AVComposerDefine.h

4.4 API Definition

The API definition is depicted here.

VIVOTEK CONFIDENTIAL
2017.01.20

4.4.1 AVComposer_CreateHandle

This function creates the AVComposer object and returns an interface to it. You must call this function or [AVComposer_CreateHandleW](#) before using this module.

Syntax

```
SCODE AVComposer_CreateHandle (  
  
    AVComposer                **ppAVComposer,  
    DWORD                    dwMaxFileSize,  
    DWORD                    dwMaxTimePeriod,  
    AVCOMPOSER\_PF\_REQUEST\_FILENAME pfRequestFilename,  
    HANDLE                   hContext  
);
```

Parameters

****ppAVComposer**

[out] Pointer to an AVComposer interface.

dwMaxFileSize

[in] Maximum file size, in bytes.

dwMaxTimePeriod

[in] Maximum file duration, in seconds.

pfRequestFilename

[in] Pointer to a callback function which is used to get new filename.

hContext

[in] An instance which is associated with the callback function.

Return Values

S_OK

Create the module successfully.

S_FAIL

Fail to create the module.

Remarks

Applications can use dwMaxFileSize to limit per media file size. When file size reaches dwMaxFileSize, AVComposer closes previous file and calls the callback function before opening a new media file. On the other hand, Applications can use dwMaxTimePeriod to limit per media file duration. When file duration reaches dwMaxTimePeriod, AVComposer closes previous file and calls the callback function before opening a new media file.

Requirements

AVComposer.h

See Also

4.4.2 AVComposer_CreateHandleW

This function creates the AVComposer object and returns an interface to it. You must call this function or [AVComposer_CreateHandle](#) before using this module.

Syntax

```

SCOPE AVComposer_CreateHandleW (
    AVComposer                **ppAVComposer,
    DWORD                    dwMaxFileSize,
    DWORD                    dwMaxTimePeriod,
    AVCOMPOSER\_PF\_REQUEST\_FILENAMEW pfRequestFilenameW,
    HANDLE                    hContext
);

```

Parameters

****ppAVComposer**

[out] Pointer to an AVComposer interface.

dwMaxFileSize

[in] Maximum file size, in bytes.

dwMaxTimePeriod

[in] Maximum file duration, in seconds.

pfRequestFilenameW

[in] Pointer to a callback function which is used to get new filename.

hContext

[in] An instance which is associated with the callback function.

Return Values

S_OK

Create the module successfully.

S_FAIL

Fail to create the module.

Remarks

Applications can use dwMaxFileSize to limit per media file size. When file size reaches dwMaxFileSize, AVComposer closes previous file and calls the callback function before opening a new media file. On the other hand, Applications can use dwMaxTimePeriod to limit per media file duration. When file duration reaches dwMaxTimePeriod, AVComposer closes previous file and calls the callback function before opening a new media file.

Requirements

AVComposer.h

See Also

4.4.3 AVComposer_ReleaseHandle

Call this function to release the AVComposer object.

Syntax

```
SCODE AVComposer_ReleaseHandle (  
  
                                AVComposer          *pAVComposer  
  
);
```

Parameters

***pAVComposer**

[in] An AVComposer interface which is returned by [AVComposer_CreateHandle](#) or [AVComposer_CreateHandleW](#).

Return Values

S_OK

Release the object successfully.

S_FAIL

Fail to release the object.

Remarks

Requirements

AVComposer.h

See Also

4.4.4 AVComposer_StartRecording

Start AVComposer recording function.

Syntax

```
SCODE AVComposer_StartRecording (  
  
    AVComposer          *pAVComposer,  
    char                *pszFilename,  
    DWORD               dwOptions  
);
```

Parameters

***pAVComposer**

[in] An AVComposer interface which is returned by [AVComposer_CreateHandle](#) or [AVComposer_CreateHandleW](#).

***pszFilename**

[in] Filename. This is for the first media file.

dwOptions

[in] One of record options which is defined in [EAVCOMPOSER_RECORD_OPTIONS](#). The only valid value of this parameter is eRecOptVivoTrack.

Return Values

S_OK

Start recording function successfully.

S_FAIL

Fail to start recording function.

Remarks

Requirements

AVComposer.h

[See Also](#)

VIVOTEK CONFIDENTIAL
2017.01.20

4.4.5 AVComposer_StopRecording

Stop AVComposer recording function.

Syntax

```
SCODE AVComposer_StopRecording (  
  
                                AVComposer          *pAVComposer  
  
);
```

Parameters

***pAVComposer**

[in] An AVComposer interface which is returned by [AVComposer_CreateHandle](#) or [AVComposer_CreateHandleW](#).

Return Values

S_OK

Stop recording function successfully.

S_FAIL

Fail to stop recording function.

E_AVCOMPOSER_RECORD_TIMESTAMP_INVALID

One or more recorded media packets have non-incremental timestamp.

E_AVCOMPOSER_RECORD_PARSE_CI_FAIL

One or more recorded media packets have invalid codec information.

Remarks

Requirements

AVComposer.h

See Also

VIVOTEK CONFIDENTIAL
2017.01.20

4.4.6 AVComposer_InputPacket

Write an input media packet to file.

Syntax

```
SCODE AVComposer_InputPacket (
    AVComposer          *pAVComposer,
    TMediaDataPacketInfoV3 *ptMediaPacket,
    DWORD               *pdwFileSize
);
```

Parameters

***pAVComposer**

[in] An AVComposer interface which is returned by [AVComposer_CreateHandle](#) or [AVComposer_CreateHandleW](#).

***ptMediaPacket**

[in] A pointer to a media packet.

***pdwFileSize**

[out] A Pointer to a value which is used to receive current file size.

Return Values

S_OK

Write an input packet successfully.

S_FAIL

Fail to write the input packet. For example, if the input media packet's codec type is different from previous one, then this function fails and returns S_FAIL.

E_AVCOMPOSER_RECORD_CODEC_NOT_SUPPORTED

AVComposer doesn't support the input media packet's codec type.

E_AVCOMPOSER_RECORD_TIMESTAMP_INVALID

The input media packet has non-incremental timestamp.

E_AVCOMPOSER_RECORD_PARSE_CI_FAIL

The input media packet has invalid codec information.

Remarks

Requirements

AVComposer.h

4.4.7 AVComposer_SetAuthenticity

Set authenticity password.

Syntax

```
SCODE AVComposer_SetAuthenticity (  
  
                                AVComposer      *pAVComposer,  
                                char              *pszPassword  
  
);
```

Parameters

***pAVComposer**

[in] An AVComposer interface which is returned by [AVComposer_CreateHandle](#) or [AVComposer_CreateHandleW](#).

***pszPassword**

[in] Password. This is used while generating VIVOTEK proprietary watermark.

Return Values

S_OK

Set the password successfully.

S_FAIL

Fail to set password.

Remarks

Requirements

AVComposer.h

See Also
