
Juphoon Protocol Framework

Real-time Transport Protocol

Published: Jan 2006

For more information on Juphoon Protocol Framework, see <http://www.juphoon.com>

Juphoon RTP Function Definition

Juphoon System Software Corporation.

<http://www.juphoon.com>

Tel: +86-574-87287820

Fax: +86-574-87304379

Text Part Number: 101-007-01-01

Copyright © 2007, Juphoon System Software Corporation

All rights reserved.

Contents

1. INTRODUCTION.....	5
1.1 PURPOSE.....	5
1.2 AUDIENCE	5
1.3 SCOPE.....	5
1.4 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS	5
2. SYSTEM ENVIRONMENT.....	6
2.1 BASIC DATA TYPES	6
2.2 PLATFORM TYPES	6
2.3 COMMON CONCEPTS	7
2.3.1 <i>Module ID</i>	7
2.3.2 <i>Instance ID</i>	7
2.3.3 <i>Task ID</i>	7
2.3.4 <i>Processor ID</i>	8
3. USER INTERFACES	8
3.1 INTERFACE STRUCTURES	8
3.1.1 <i>EN_RTP_PAYLOAD_TYPE</i>	8
3.1.2 <i>ST_RTP_RTP_INFO</i>	10
3.1.3 <i>ST_RTP_RTCP_INFO</i>	10
3.1.4 <i>ST_ZOS_INET_ADDR</i>	11
3.1.5 <i>ST_ZOS_INET_IP</i>	12
3.1.6 <i>PFN_RTPDRTP</i>	12
3.1.7 <i>PFN_RTPDRTCPAPP</i>	12
3.2 CONFIG INTERFACES.....	12
3.2.1 <i>Rtp_CfgGetSessTaskPriority</i>	13
3.2.2 <i>Rtp_CfgGetSessTaskStackSize</i>	13
3.2.3 <i>Rtp_CfgGetSessTaskQueueSize</i>	13
3.2.4 <i>Rtp_CfgGetTptTaskPriority</i>	14
3.2.5 <i>Rtp_CfgGetTptTaskStackSize</i>	14
3.2.6 <i>Rtp_CfgGetTptSelectTimeout</i>	14
3.2.7 <i>Rtp_CfgGetIpTosVal</i>	15
3.2.8 <i>Rtp_CfgGetLogLevel</i>	15
3.2.9 <i>Rtp_CfgGetSessNum</i>	16
3.2.10 <i>Rtp_CfgGetPtpNum</i>	16
3.2.11 <i>Rtp_CfgGetSenderNum</i>	16
3.2.12 <i>Rtp_CfgGetPortBegin</i>	17
3.2.13 <i>Rtp_CfgSetSessTaskPriority</i>	17
3.2.14 <i>Rtp_CfgSetSessTaskStackSize</i>	18
3.2.15 <i>Rtp_CfgSetSessTaskQueueSize</i>	18
3.2.16 <i>Rtp_CfgSetTptTaskPriority</i>	18
3.2.17 <i>Rtp_CfgsetTptTaskStackSize</i>	19

3.2.18	<i>Rtp_CfgSetTptSelectTimeout</i>	19
3.2.19	<i>Rtp_CfgSetIpTosVal</i>	20
3.2.20	<i>Rtp_CfgSetLogLevel</i>	20
3.2.21	<i>Rtp_CfgSetSessNum</i>	20
3.2.22	<i>Rtp_CfgsetPtptNum</i>	21
3.2.23	<i>Rtp_CfgSetSenderNum</i>	21
3.2.24	<i>Rtp_CfgSetPortBegin</i>	22
3.3	TASK INTERFACES	22
3.3.1	<i>Rtp_Start</i>	22
3.3.2	<i>Rtp_Stop</i>	22
3.4	USERFUL INTERFACES	23
3.4.1	<i>Rtp_Open</i>	23
3.4.2	<i>Rtp_OpenX</i>	24
3.4.3	<i>Rtp_Close</i>	24
3.4.4	<i>Rtp_RtcpOpen</i>	25
3.4.5	<i>Rtp_RtcpOpenX</i>	25
3.4.6	<i>Rtp_RtcpEnable</i>	26
3.4.7	<i>Rtp_RtpSend</i>	26
3.4.8	<i>Rtp_RtpSendX</i>	27
3.4.9	<i>Rtp_RtpSendM</i>	27
3.4.10	<i>Rtp_RtpSendT</i>	28
3.4.11	<i>Rtp_RtcpAppSend</i>	28
3.4.12	<i>Rtp_RtcpAppResend</i>	29
3.4.13	<i>Rtp_SetRmtAddr</i>	29
3.4.14	<i>Rtp_SetPayload</i>	30
3.4.15	<i>Rtp_SetProf</i>	30
3.4.16	<i>Rtp_SetRdRtcpApp</i>	31
3.4.17	<i>Rtp_GetSsrc</i>	32
3.5	VERSION INTERFACES	32
3.5.1	<i>Rtp_GetVersion</i>	32
4.	USAGE EXPLANATION	32

List of Tables

TABLE 2-1	BASIC DATA TYPES	6
TABLE 2-2	PLATFORM TYPES	7
TABLE 3-1	EN_RTP_PAYLOAD_TYPE	10
TABLE 3-2	ST_RTP_RTP_INFO	10
TABLE 3-3	ST_RTP_RTCP_INFO	11
TABLE 3-5	ST_ZOS_INET_ADDR	11
TABLE 3-6	ST_ZOS_INET_IP	12
TABLE 4-1:	CALLING SEQUENCE FOR RTP	33

List of Figures

FIGURE 2-1 RTP STACK FRAMEWORK6

1. Introduction

RTP, the real-time transport protocol, provides end-to-end network transport functions suitable for applications transmitting real-time data, such as audio, video or simulation data, over multicast or unicast network services. The data transport is augmented by a control protocol (RTCP) to allow monitoring of the data delivery in a manner scalable to large multicast networks, and to provide minimal control and identification functionality. RTP and RTCP are designed to be independent of the underlying transport and network layers.

This document describes the function definitions of Protocol RTP/RTCP and the usage of them. The examples in the document are very simple for explaining purpose, which can not be used for business projects directly and are not ensured to work at all situations.

1.1 Purpose

This document is provided to developers who will develop their own software with the functions of Protocol RTP/RTCP.

1.2 Audience

The readers of this document are assumed to have a working knowledge of Protocol RTP/RTCP.

1.3 Scope

This document provides functions' definitions of Protocol RTP/RTCP and their usages but not the design and implementation of the protocol.

1.4 Definitions, Acronyms, and Abbreviations

The following definitions, acronyms, and abbreviations are used in this document:

Abbreviation	Description
RTP	Real-Time Transport Protocol
RTCP	Real-Time Transport Control Protocol
ZOS	Zero Operating System

2. System environment

This section describes the environment in which RTP is designed to operate. Figure 2-1 illustrates the RTP stack framework.

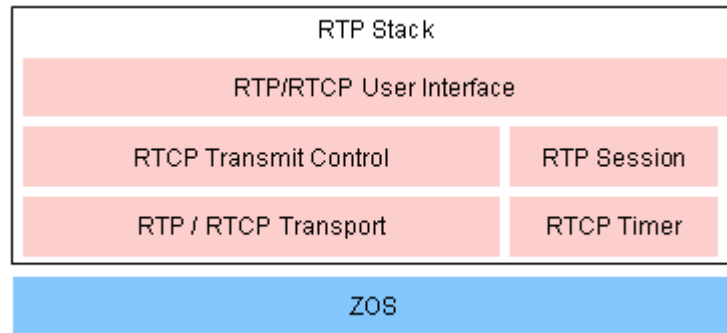


Figure 2-1 RTP Stack Framework

2.1 Basic data types

There are some basic data types provided by ZOS platform. Table 2-1 lists these types used by the RTP Stack.

Name	Type
ZDOUBLE	double
ZFLOAT	float
ZLONG	long
ZINT	int
ZSHORT	short
ZCHAR	char
ZULONG	unsigned long
ZUINT	unsigned int
ZSIZE_T	unsigned int
ZUSHORT	unsigned short
ZUCHAR	unsigned char
ZBOOL	int
ZVOID	void

Table 2-1 Basic Data Types

2.2 Platform types

Table 2-2 lists the basic platform types.

Name	Type
ZMUTEX	Mutex
ZSEM	Semaphore
ZTIME_T	Time
ZFUNCPTR	Function Pointer
ZVOIDFUNCPTR	Void Function Pointer
ZLOGID	Log ID
ZMODID	Module ID
ZINSTID	Instance ID
ZTASKID	Task ID
ZTIMERID	Timer ID
ZEVENTID	Event ID
ZPOOLID	Pool ID

Table 2-2 Platform Types

2.3 Common concepts

This section describes four kinds of common ID structures.

2.3.1 Module ID

In the RTP architecture, each module is assigned a unique ID known as the module ID. The module ID is a 16-bit unsigned integer. There are two kinds of modules, the ZOS basic modules and a module defined by users. The ZOS basic modules are listed as the following:

- system module
- SIP module
- RTP module
- test module

Note that the connection manager module is defined by users.

2.3.2 Instance ID

An instance of a specific module is assigned an ID, too. There may be several instances share one module. The instance ID is used to distinguish multiple instances. Also as the module ID, the instance ID is a 16-bit unsigned integer, too.

2.3.3 Task ID

Task ID is used to identify different tasks. A task ID is a 32-bit unsigned integer. The leftmost 16 bits of a task ID are used to store a module ID and the other 16 bits are used to store an instance ID of the module. There are five major tasks which are listed as the following:

- Timer task

- Log task (which is optional)
- SIP task
- RTP task
- SUA task

2.3.4 Processor ID

An operating system may be supported by multiple processors. Each processor has an ID known as the processor ID to distinguish multiple processors when they are communicating with each other. At present, the distributed operating system is not supported.

3. User interfaces

3.1 Interface structures

3.1.1 EN_RTP_PAYLOAD_TYPE

It defines the payload type supported by Juphoon RTP stack now.

```

/* RTP payload type */
typedef enum EN_RTP_PAYLOAD_TYPE
{
    EN_RTP_PAYLOAD_PCMU = 0,
    EN_RTP_PAYLOAD_GSM = 3,
    EN_RTP_PAYLOAD_G723 = 4,
    EN_RTP_PAYLOAD_DVI4_8K = 5,
    EN_RTP_PAYLOAD_DVI4_16K = 6,
    EN_RTP_PAYLOAD_LPC = 7,
    EN_RTP_PAYLOAD_PCMA = 8,
    EN_RTP_PAYLOAD_G722 = 9,
    EN_RTP_PAYLOAD_L16_STEREO = 10,
    EN_RTP_PAYLOAD_L16_MONO = 11,
    EN_RTP_PAYLOAD_QCELP = 12,
    EN_RTP_PAYLOAD_CN = 13,
    EN_RTP_PAYLOAD_MPA = 14,
    EN_RTP_PAYLOAD_G728 = 15,
    EN_RTP_PAYLOAD_DVI4_11K = 16,
    EN_RTP_PAYLOAD_DVI4_22K = 17,
    EN_RTP_PAYLOAD_G729 = 18,
    EN_RTP_PAYLOAD_CELB = 25,
    EN_RTP_PAYLOAD_JPEG = 26,
    EN_RTP_PAYLOAD_NV = 28,
    EN_RTP_PAYLOAD_H261 = 31,
    EN_RTP_PAYLOAD_MPV = 32,
    EN_RTP_PAYLOAD_MP2T = 33,
    EN_RTP_PAYLOAD_H263 = 34,
    EN_RTP_PAYLOAD_MAX = 128
} EN_RTP_PAYLOAD_TYPE;

```

All the constant integer values in the enum above are defined in RFC 3551.

Constant	Description
EN_RTP_PAYLOAD_PCMU	PCMU. Its value is 0.
EN_RTP_PAYLOAD_GSM	GSM. Its value is 3.
EN_RTP_PAYLOAD_G723	G723. Its value is 4.
EN_RTP_PAYLOAD_PCMA	PCMA. Its value is 8.
EN_RTP_PAYLOAD_G722	G722. Its value is 9.
EN_RTP_PAYLOAD_G728	G728. Its value is 15.
EN_RTP_PAYLOAD_G729	G729. Its value is 18.
EN_RTP_PAYLOAD_JPEG	JPEG. Its value is 26.
EN_RTP_PAYLOAD_H261	H261. Its value is 31.

EN_RTP_PAYLOAD_H263	H263. Its value is 34.
EN_RTP_PAYLOAD_MPEG4	MPEG4. Its value is 99.
EN_RTP_PAYLOAD_MAX	Maximum number of payload types. Its value is 128.

Table 3-1 EN_RTP_PAYLOAD_TYPE

3.1.2 ST_RTP_RTP_INFO

It is the RTP information structure defined in rtp_type.h.

```

/* rtp packet information */
typedef struct tagRTP_RTP_INFO
{
    ZUCHAR ucVer;           /* rtp version */
    ZUCHAR ucPadding;      /* padding flag */
    ZUCHAR ucExt;          /* extension exist flag */
    ZUCHAR ucCsrcCount;    /* contributing source count */
    ZUCHAR ucMarker;       /* marker flag */
    ZUCHAR ucPayload;      /* payload type */
    ZUSHORT wSeq;          /* sequence number */
    ZUINT iTs;             /* timestamp */
    ZUINT iSsrc;           /* synchronization source */
    ZUINT aiCsrc[15];      /* contributing source list */
} ST_RTP_RTP_INFO;

```

Its members are described below:

Member	Description
ucVer	RTP version
ucPadding	Padding flag
ucExt	Extension exist flag
ucCsrcCount	Contributing source count
ucMarker	Marker flag
ucPayload	Payload type
wSeq	Sequence number
iTs	Timestamp
iSsrc	Synchronization source
aiCsrc	Contributing source list

Table 3-2 ST_RTP_RTP_INFO

3.1.3 ST_RTP_RTCP_INFO

It is the user information structure defined in rtp_type.h.

```
typedef struct tagRTP_RTCP_INFO
{
    ZUCHAR ucSubType;           /* subtype */
    ZUCHAR aucSpare[3];        /* for 32 bit alignment */
    ZUINT iSsrc;               /* synchronization source */
} ST_RTP_RTCP_INFO;
```

Its members are described below:

Member	Description
ucSubType	Indicates the subtype of RTCP packet.
iSsrc	Synchronization source

Table 3-3 ST_RTP_RTCP_INFO

3.1.4 ST_ZOS_INET_ADDR

It is the ZOS network address structure defined in zos_inet.h.

```
typedef struct tagZOS_INET_ADDR
{
    ZUCHAR ucType;             /* ZINET_IPV4... */
    ZUCHAR aucSpare[1];        /* for 32 bit alignment */
    ZUSHORT wPort;             /* not order[host or n/w] dependent */
    union
    {
        ZULONG dwIp;           /* not order[host or n/w] dependent */
        ZUCHAR aucIp[ZINET_IPV4_ADDR_SIZE]; /* ipv4 address */
        ZUCHAR aucIpv6[ZINET_IPV6_ADDR_SIZE]; /* ipv6 address */
    } u;
} ST_ZOS_INET_ADDR;
```

Its members are described below:

Member	Description
ucType	Network address type, Its value can be ZINET_IPV4 or ZINET_IPV6.
wPort	A port number which is order independent.
u.dwIp	An IPv4 address which is order independent.
u.aucIp	An array used to store the IPv4 address (u.dwIp , consisting of 4 bytes), with each element holding one byte of the IPv4 address.
u. aucIpv6	An array used to store the IPv6 address.

Table 3-5 ST_ZOS_INET_ADDR

3.1.5 ST_ZOS_INET_IP

It is the ZOS IP address structure defined in zos_inet.h.

```
typedef struct tagZOS_INET_IP
{
    ZUCHAR ucType;                /* ZINET_IPV4... */
    ZUCHAR aucSpare[3];          /* for 32 bit alignment */
    union
    {
        ZULONG dwIp;             /* not order[host or n/w] dependent */
        ZUCHAR aucIp[ZINET_IPV4_ADDR_SIZE]; /* ipv4 address */
        ZUCHAR aucIpv6[ZINET_IPV6_ADDR_SIZE]; /* ipv6 address */
    } u;
} ST_ZOS_INET_IP;
```

Its members are described below:

Member	Description
ucType	Network address type, Its value can be ZINET_IPV4 or ZINET_IPV6.
u.dwIp	An IPv4 address which is order independent.
u.aucIp	An array used to store the IPv4 address (u.dwIp , consisting of 4 bytes), with each element holding one byte of the IPv4 address.
u.aucIpv6	An array used to store the IPv6 address.

Table 3-6 ST_ZOS_INET_IP

3.1.6 PFN_RTPDRTP

The prototype of callback function used when RTP packet received.

```
typedef ZINT (*PFN_RTPDRTP)(ZULONG dwUserId, ZULONG dwRtpId, \
    ST\_ZOS\_INET\_ADDR *pstRmtAddr, ZUCHAR *pucData, \
    ZUINT iDataLen, ST\_RTP\_RTP\_INFO *pstInfo);
```

3.1.7 PFN_RTPDRTCPAPP

The prototype of callback function used when RTCP packet in APP format received.

```
typedef ZINT (*PFN_RTPDRTCPAPP)(ZULONG dwUserId, ZULONG dwRtpId, \
    ST\_RTP\_RTCP\_INFO *pstInfo, ZUCHAR *pucAppData, ZUINT iAppLen);
```

3.2 Config Interfaces

These interfaces are included in rtp_cfg.h.

3.2.1 Rtp_CfgGetSessTaskPriority

Gets the session task priority. The default priority is ZTASK_PRIORITY_NORMAL.

```
ZINT Rtp_CfgGetSessTaskPriority(ZINT *piPriority);
```

[Parameters]

Input parameters:

None.

Output parameters:

ZINT *piPriority

The task priority.

[Return value]

Returns ZOK.

3.2.2 Rtp_CfgGetSessTaskStackSize

Gets the session task stack size. The default size is 16K.

```
ZINT Rtp_CfgGetSessTaskStackSize(ZULONG *pdwStackSize);
```

[Parameters]

Input parameters:

None.

Output parameters:

ZULONG *pdwStackSize

The stack size.

[Return value]

Returns ZOK.

3.2.3 Rtp_CfgGetSessTaskQueueSize

Gets the session task queue size. The default size is 100.

```
ZINT Rtp_CfgGetSessTaskQueueSize(ZULONG *pdwQueueSize);
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZULONG *pdwQueueSize
```

The queue size.

[Return value]

Returns ZOK.

3.2.4 Rtp_CfgGetTptTaskPriority

Gets the transport task priority. The default priority is ZTASK_PRIORITY_NORMAL.

```
ZINT Rtp_CfgGetTptTaskPriority(ZINT *piPriority);
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZINT *piPriority
```

The task priority.

[Return value]

Returns ZOK.

3.2.5 Rtp_CfgGetTptTaskStackSize

Gets the transport task stack size. The default size is 16K.

```
ZINT Rtp_CfgGetTptTaskStackSize(ZULONG *pdwStackSize);
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZULONG *pdwStackSize
```

The stack size.

[Return value]

Returns ZOK.

3.2.6 Rtp_CfgGetTptSelectTimeout

Gets the transport select timeout. The default value is 1000ms.

```
ZINT Rtp_CfgGetTptSelectTimeout(ZINT *piSelectTimeout);
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZINT *piSelectTimeout
```

The select timeout.

[Return value]

Returns ZOK.

3.2.7 Rtp_CfgGetIpTosVal

Get the TOS settings of RTP.

```
ZINT Rtp_CfgGetIpTosVal(ZUCHAR *pucPrecLevel, ZUCHAR *pucTosType);
```

[Parameters]

Input parameters:

None.

Output parameters:

```
ZUCHAR *pucPrecLevel
```

Point to the precedence value.

```
ZUCHAR *pucTosType
```

Point to the TOS type.

[Return value]

Returns ZOK.

3.2.8 Rtp_CfgGetLogLevel

Gets the log level. The default level is ZLOG_LEVEL_ERROR.

```
ZINT Rtp_CfgGetLogLevel(ZULONG *pdwLevel);
```

[Parameters]

Input parameters:

None.

Output parameters:

ZULONG *pdwLevel

The log level.

[Return value]

Returns ZOK.

3.2.9 Rtp_CfgGetSessNum

Gets the number of sessions. The default number is 20.

```
ZINT Rtp_CfgGetSessNum(ZULONG *pdwNum);
```

[Parameters]

Input parameters:

None.

Output parameters:

ZULONG *pdwNum

The number of sessions.

[Return value]

Returns ZOK.

3.2.10 Rtp_CfgGetPtptNum

Gets the number of participants. The default number is 30.

```
ZINT Rtp_CfgGetPtptNum(ZULONG *pdwNum);
```

[Parameters]

Input parameters:

None.

Output parameters:

ZULONG *pdwNum

The number of participants.

[Return value]

Returns ZOK.

3.2.11 Rtp_CfgGetSenderNum

Gets the number of senders. The default number is 30.

```
ZINT Rtp_CfgGetSenderNum (ZULONG *pdwNum);
```

[Parameters]

Input parameters:

None.

Output parameters:

ZULONG *pdwNum

The number of senders.

[Return value]

Returns ZOK.

3.2.12 Rtp_CfgGetPortBegin

Gets the beginning port. The default beginning port is 3700.

```
ZINT Rtp_CfgGetPortBegin (ZULONG *pdwPort);
```

[Parameters]

Input parameters:

None.

Output parameters:

ZULONG *pdwPort

The beginning port.

[Return value]

Returns ZOK.

3.2.13 Rtp_CfgSetSessTaskPriority

Sets the session task priority.

```
ZINT Rtp_CfgSetSessTaskPriority (ZINT iPriority);
```

[Parameters]

Input parameters:

ZINT iPriority

The session task priority.

Output parameters:

None.

[Return value]

Returns ZOK.

3.2.14 Rtp_CfgSetSessTaskStackSize

Sets the session task size.

```
ZINT Rtp_CfgSetSessTaskStackSize(ZULONG dwStackSize);
```

[Parameters]**Input parameters:**

ZULONG dwStackSize

The stack size.

Output parameters:

None.

[Return value]

Returns ZOK.

3.2.15 Rtp_CfgSetSessTaskQueueSize

Sets the session task queue size.

```
ZINT Rtp_CfgSetSessTaskQueueSize(ZULONG dwQueueSize);
```

[Parameters]**Input parameters:**

ZULONG dwQueueSize

The session task queue size.

Output parameters:

None.

[Return value]

Returns ZOK.

3.2.16 Rtp_CfgSetTptTaskPriority

Sets the transport task priority.

```
ZINT Rtp_CfgSetTptTaskPriority(ZINT iPriority);
```

[Parameters]

Input parameters:

ZINT iPriority
The transport task priority.

Output parameters:

None.

[Return value]

Returns ZOK.

3.2.17 Rtp_CfgsetTptTaskStackSize

Sets the transport task stack size.

```
ZINT Rtp_CfgsetTptTaskStackSize(ZULONG dwStackSize);
```

[Parameters]**Input parameters:**

ZULONG dwStackSize
The transport task stack size.

Output parameters:

None.

[Return value]

Returns ZOK.

3.2.18 Rtp_CfgSetTptSelectTimeout

Sets the transport select timeout.

```
ZINT Rtp_CfgSetTptSelectTimeout(ZINT iSelectTimeout);
```

[Parameters]**Input parameters:**

ZINT iSelectTimeout
The select timeout.

Output parameters:

None.

[Return value]

Returns ZOK.

3.2.19 Rtp_CfgSetIpTosVal

Set the TOS settings of RTP.

```
ZINT Rtp_CfgSetIpTosVal(ZUCHAR ucPrecLevel, ZUCHAR ucTosType);
```

[Parameters]

Input parameters:

ZUCHAR ucPrecLevel

The precedence value from 0 to 7.

ZUCHAR ucTosType

The TOS type which could be any combination of ZSOCK_TOS_XXX.

Output parameters:

None.

[Return value]

Returns ZOK.

3.2.20 Rtp_CfgSetLogLevel

Sets the log level.

```
ZINT Rtp_CfgSetLogLevel(ZULONG dwLevel);
```

[Parameters]

Input parameters:

ZULONG dwLevel

The log level.

Output parameters:

None.

[Return value]

Returns ZOK.

3.2.21 Rtp_CfgSetSessNum

Sets the number of session.

```
ZINT Rtp_CfgSetSessNum(ZULONG dwNum);
```

[Parameters]

Input parameters:

ZULONG dwNum
The number of sessions.

Output parameters:

None.

[Return value]

Returns ZOK.

3.2.22 Rtp_CfgsetPtptNum

Sets the number of participants.

```
ZINT Rtp_CfgsetPtptNum(ZULONG dwNum);
```

[Parameters]**Input parameters:**

ZULONG dwNum
The number of participants.

Output parameters:

None.

[Return value]

Returns ZOK.

3.2.23 Rtp_CfgSetSenderNum

Sets the number of senders.

```
ZINT Rtp_CfgSetSenderNum(ZULONG dwNum);
```

[Parameters]**Input parameters:**

ZULONG dwNum
The number of senders.

Output parameters:

None.

[Return value]

Returns ZOK.

3.2.24 Rtp_CfgSetPortBegin

Sets the beginning port.

```
ZINT Rtp_CfgSetPortBegin(ZULONG dwPort);
```

[Parameters]

Input parameters:

ZULONG dwPort

The beginning port.

Output parameters:

None.

[Return value]

Returns ZOK.

3.3 Task Interfaces

These interfaces are included in rtp_task.h.

3.3.1 Rtp_Start

This is a task interface used to start the RTP module.

```
ZINT Rtp_Start();
```

[Parameters]

Input parameters:

None.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.3.2 Rtp_Stop

Stops the RTP module.

```
ZVOID Rtp_Stop();
```

[Parameters]

Input parameters:

None.

Output parameters:

None.

[Return value]

None.

3.4 Useful Interfaces

Here are upper interfaces provided to users. These interfaces are included in rtp_ui.h.

3.4.1 Rtp_Open

This function is used to open an RTP channel.

```
ZINT Rtp_Open(ST\_ZOS\_INET\_IP *pstLocalIp, ZUCHAR ucPayload,
              ZULONG dwUserId, PFN\_RTDRTP pfnRdRtp,
              ZUSHORT *pwRtpPort, ZULONG *pdwRtpId);
```

[Parameters]

Input parameters:

[ST_ZOS_INET_IP](#) *pstLocalIp

The local IP.

ZUCHAR ucPayload

Indicates the payload type. For detailed information please refer to [EN RTP PAYLOAD TYPE](#).

ZULONG dwUserId

The user ID.

[PFN_RTDRTP](#) pfnRdRtp

The RTP data callback function.

Output parameters:

ZUSHORT *pwRtpPort

A pointer to the opened RTP port as the result of opening a channel.

ZULONG *pdwRtpId

A pointer to the opened RTP id as the result of opening a channel.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.4.2 Rtp_OpenX

This function is used to open an RTP channel with a specific port.

```
ZINT Rtp_OpenX(ST\_ZOS\_INET\_IP *pstLocalIp, ZUCHAR ucPayload,
              ZULONG dwUserId, PFN RTPDRTP pfnRdRtp,
              ZUSHORT wRtpPort, ZULONG *pdwRtpId)
```

[Parameters]

Input parameters:

[ST_ZOS_INET_IP](#) *pstLocalIp

The local IP

ZUCHAR ucPayload

Indicates the payload type. For detailed information please refer to [EN RTP_PAYLOAD_TYPE](#).

ZULONG dwUserId

The user ID.

[PFN RTPDRTP](#) pfnRdRtp

The callback function.

ZUSHORT wRtpPort

A port as the RTP port number of opening a channel.

Output parameters:

ZULONG *pdwRtpId

A pointer to the opened RTP ID as the result of opening a channel.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.4.3 Rtp_Close

This function is used to close an RTP channel.

```
ZINT Rtp_Close(ZULONG dwRtpId);
```

[Parameters]

Input parameters:

ZULONG dwRtpId

Indicates the RTP channel which is to be closed. If no RTP channel could be found based on the RTP id, ZFAILED will be returned.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.4.4 Rtp_RtcpOpen

This function is used to open an RTCP channel.

```
ZINT Rtp_RtcpOpen(ZULONG dwRtpId, ZUSHORT *pwRtcpPort);
```

[Parameters]**Input parameters:**

ZULONG dwRtpId

A RTP channel ID.

Output parameters:

ZUSHORT *pwRtcpPort

A pointer to the opened RTCP port as the result of opening a channel.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.4.5 Rtp_RtcpOpenX

This function is used to open an RTCP channel with a specific port.

```
ZINT Rtp_RtcpOpenX(ZULONG dwRtpId, ZUSHORT wRtcpPort);
```

[Parameters]**Input parameters:**

ZULONG dwRtpId

A RTP channel ID.

ZUSHORT wRtcpPort

A port as the RTCP port number of a RTP channel.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.4.6 Rtp_RtcpEnable

This function is used to enable or disable RTCP in RTP channel.

```
ZINT Rtp_RtcpEnable(ZULONG dwRtpId, ZBOOL bEnable)
```

[Parameters]

Input parameters:

```
ZULONG dwRtpId
```

A RTP channel ID.

```
ZBOOL bEnable
```

A Boolean value indicates to enable or disable RTCP.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.4.7 Rtp_RtpSend

This function is used to send RTP data.

```
ZINT Rtp_RtpSend(ZULONG dwRtpId, ZUCHAR *pucData, ZUINT iLen);
```

[Parameters]

Input parameters:

```
ZULONG dwRtpId
```

A RTP ID through which data are sent. If the id is invalid, ZFAILED will be returned.

```
ZUCHAR *pucData
```

The data which are to be sent.

```
ZUINT iLen
```

Length of the data.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.4.8 Rtp_RtpSendX

This function is used to send RTP data.

```
ZINT Rtp_RtpSendX(ZULONG dwRtpId, ZUCHAR *pucData, ZUINT iLen
                 ZUINT iRepeatTimes);
```

[Parameters]

Input parameters:

ZULONG dwRtpId

A RTP ID through which data are sent. If the id is invalid, ZFAILED will be returned.

ZUCHAR *pucData

The data which are to be sent.

ZUINT iLen

Length of the data.

ZUINT iRepeatTimes

The repeat times of sending.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.4.9 Rtp_RtpSendM

This function is used to send RTP data with marker bits.

```
ZINT Rtp_RtpSendM(ZULONG dwRtpId, ZUCHAR *pucData, ZUINT iLen);
```

[Parameters]

Input parameters:

ZULONG dwRtpId

A RTP ID through which data are sent. If the id is invalid, ZFAILED will be returned.

ZUCHAR *pucData

The data which are to be sent.

ZUINT iLen

Length of the data.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.4.10 Rtp_RtpSendT

This function is used to send RTP data with marker bits.

```
ZINT Rtp_RtpSendT(ZULONG dwRtpId, ZUCHAR *pucData, ZUINT iLen
                  ZUINT iTsInc, ZUINT iRepeatTimes);
```

[Parameters]**Input parameters:**

ZULONG dwRtpId

A RTP ID through which data are sent. If the id is invalid, ZFAILED will be returned.

ZUCHAR *pucData

The data which are to be sent.

ZUINT iLen

Length of the data.

ZUINT iTsInc

Increase value of timestamp.

ZUINT iRepeatTimes

The repeat times of sending.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.4.11 Rtp_RtcpAppSend

This function is used to send RTCP APP data through a RTP channel.

```
ZINT Rtp_RtcpAppSend(ZULONG dwRtpId, ZUCHAR ucSubType,
                     ZUCHAR *pucData, ZUINT iLen);
```

[Parameters]

Input parameters:

ZULONG dwRtpId

A RTP ID through which data are sent. If the id is invalid, ZFAILED will be returned.

ZUCHAR ucSubType

Indicates the subtype of RTCP APP packet.

ZUCHAR *pucData

Points to the RTCP APP data.

ZUINT iLen

Length of RTCP APP data.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.4.12 Rtp_RtcpAppResend

This function is used to resend RTCP APP data through a RTP channel.

```
ZINT Rtp_RtcpAppResend(ZULONG dwRtpId);
```

[Parameters]**Input parameters:**

ZULONG dwRtpId

A RTP ID through which data are sent. If the id is invalid, ZFAILED will be returned.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.4.13 Rtp_SetRmtAddr

This function is used to set the remote address.

```
ZINT Rtp_SetRmtAddr(ZULONG dwRtpId, ST\_ZOS\_INET\_ADDR *pstRmtAddr);
```

[Parameters]**Input parameters:**

ZULONG dwRtpId

The RTP id indicates the RTP channel on which the remote address will be set.

[.ST_ZOS_INET_ADDR](#) *pstRmtAddr

The remote address to be set on the intended RTP channel.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.4.14 Rtp_SetPayload

This function is used to set the payload type of an RTP channel.

```
ZINT Rtp_SetPayload(ZULONG dwRtpId, ZUCHAR ucPayload);
```

[Parameters]

Input parameters:

ZULONG dwRtpId

The RTP id indicates the RTP channel on which the payload type will be set.

ZUCHAR ucPayload

The payload type to be set on the intended RTP channel. For detailed information please refer to [EN RTP PAYLOAD TYPE](#).

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.4.15 Rtp_SetProf

This function is used to set the payload profile.

```
ZINT Rtp_SetProf(ZULONG dwRtpId, ZBOOL bEnable, ZUCHAR ucPayload,  
                ZUCHAR ucMarker, ZUCHAR ucPadding, ZUCHAR ucExt, ZUINT iTsInc);
```

[Parameters]

Input parameters:

ZULONG dwRtpId
The payload profile information.

ZBOOL bEnable
Indicates to enable or disable the profile.

ZUCHAR ucPayload
Indicates the payload number of profile.

ZUCHAR ucMarker
Indicates the marker bit.

ZUCHAR ucPadding
Indicates the padding bit.

ZUCHAR ucExt
Indicates the externsion bit.

ZUINT iTsInc
Indicates the timestamp increase.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.4.16 Rtp_SetRdRtcpApp

This function is used to set the RTCP application data read function of an RTP channel.

```
ZINT Rtp_SetRdRtcpApp(ZULONG dwRtpId, PFN\_RTDRTCPAPP pfnRdRtcpApp);
```

[Parameters]**Input parameters:**

ZULONG dwRtpId
The RTP id indicates the RTP channel on which the payload type will be set.

[PFN_RTDRTCPAPP](#) pfnRdRtcpApp
The RTCP application data callback function.

Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

3.4.17 Rtp_GetSsrc

This function is used to get the synchronization source of a RTP channel.

```
ZUINT Rtp_GetSsrc(ZULONG dwRtpId);
```

[Parameters]**Input parameters:**

ZULONG dwRtpId

The payload profile information.

Output parameters:

None.

[Return value]

The synchronization source of the RTP channel.

3.5 Version Interfaces

These interfaces are included in rtp_version.h.

3.5.1 Rtp_GetVersion

This function is used to get the RTP version.

```
ZCHAR * Rtp_GetVersion();
```

[Parameters]**Input parameters:**

None.

Output parameters:

None.

[Return value]

Returns the version string.

4. Usage Explanation

This section gives an explanation on how to use the RTP interfaces.

Interface to be called	Explains
Initialization	
Rtp_CfgInit()	Sets the default config of audio. After it returned from this function, you can customize the config of audio. If you do not want to change anything, calling this interface is not necessary.
Rtp_Start()	Performs the initialization work in the module.
Prepare for RTP Transmission	
ST_ZOS_INET_IP stLocalIp; ZULONG dwRtpId, dwIp; Zos_InetAddr("192.168.0.45", & dwIp); ZOS_IPV4_ADDR_SET(&stLocalIp, dwIp); Rtp_OpenX(&stLocalIp, 0, 0, Audio_ReadRtp, 37000, &dwRtpId);	Opens a RTP channel with specific port 37000. And the RTP callback function on receiving RTP package is set to Audio_ReadRtp.
Rtp_RtcpOpenX(dwRtpId, 37001);	Opens RTCP channel along with specific RTP channel.
Rtp_SetProf(dwRtpId, ZTRUE, 0, 0, 0, 0, 160); Rtp_SetProf(dwRtpId, ZTRUE, 3, 0, 0, 0, 160); Rtp_SetProf(dwRtpId, ZTRUE, 8, 0, 0, 0, 160);	Sets the default RTP profile configuration. The example sets 3 payload for PCMU, GSM and PCMA. This function may be also called after the capacity was negotiated.
ST_ZOS_INET_ADDR stRmtAddr; ZOS_IPV4_ADDR_SET_STR(&stRmtAddr, "192.168.0.48"); stRmtAddr.wPort = 37000; Rtp_SetRmtAddr(dwRtpId, &stRmtAddr);	Sets the remote address ip and port which will be communicated with.
Rtp_SetPayload(dwRtpId, 0);	Sets the current RTP payload type which will be sent.
Transmission	
Rtp_RtpSend(dwRtpId, pucData, iLen);	Sends data through a RTP channel. The data do not contain any RTP header.
Call the callback function.	When receiving a RTP package from a channel the module will call the callback function set in Rtp_OpenX interface.
Terminate	
Rtp_Close(dwRtpId);	Closes a RTP channel.

Table 4-1: Calling Sequence for RTP