

---

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 © 2006, Juphoon System Software Corporation.

All rights reserved.

## Contents

<b>1. INTRODUCTION.....</b>	<b>4</b>
1.1 PURPOSE.....	4
1.2 AUDIENCE .....	4
1.3 SCOPE.....	4
1.4 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS.....	4
<b>2. SYSTEM ENVIRONMENT.....</b>	<b>5</b>
2.1 BASIC DATA TYPES .....	5
2.2 PLATFORM TYPES.....	6
2.3 COMMON CONCEPTS .....	6
2.3.1 <i>Module ID</i> .....	6
2.3.2 <i>Instance ID</i> .....	6
2.3.3 <i>Task ID</i> .....	7
2.3.4 <i>Processor ID</i> .....	7
<b>3. USER INTERFACES .....</b>	<b>7</b>
3.1 INTERFACE STRUCTURES .....	7
3.1.1 <i>EN_RTP_PAYLOAD_TYPE</i> .....	7
3.1.2 <i>ST_RTP_RTP_INFO</i> .....	8
3.1.3 <i>ST_RTP_RTCP_INFO</i> .....	9
3.1.4 <i>ST_RTP_USER_INFO</i> .....	9
3.1.5 <i>PFN_RTPDATAIND</i> .....	10
3.1.6 <i>PFN_RTCPAPPIND</i> .....	10
3.1.7 <i>ST_RTP_RTP_MSG</i> .....	11
3.1.8 <i>ST_RTP_CFG</i> .....	11
3.1.9 <i>ST_ZOS_INET_ADDR</i> .....	14
3.1.10 <i>ST_ZOS_INET_IP</i> .....	14
3.2 CONFIG INTERFACES.....	15
3.2.1 <i>Rtp_CfgInit</i> .....	15
3.3 TASK INTERFACES .....	15
3.3.1 <i>Rtp_Start</i> .....	15
3.4 UPPER INTERFACES .....	16
3.4.1 <i>Rtp_Open</i> .....	16
3.4.2 <i>Rtp_OpenX</i> .....	17
3.4.3 <i>Rtp_Close</i> .....	17
3.4.4 <i>Rtp_RtcpOpen</i> .....	18
3.4.5 <i>Rtp_RtcpOpenX</i> .....	18
3.4.6 <i>Rtp_RtcpEnable</i> .....	19
3.4.7 <i>Rtp_RtpSend</i> .....	19
3.4.8 <i>Rtp_RtpSendM</i> .....	20
3.4.9 <i>Rtp_RtpResend</i> .....	21
3.4.10 <i>Rtp_RtcpAppSend</i> .....	21

3.4.11	<i>Rtp_RtcpAppResend</i> .....	22
3.4.12	<i>Rtp_SetRmtAddr</i> .....	22
3.4.13	<i>Rtp_SetPayload</i> .....	23
3.4.14	<i>Rtp_SetProf</i> .....	24
3.4.15	<i>Rtp_SetRtcpInd</i> .....	24
3.4.16	<i>Rtp_GetSsrc</i> .....	25
3.5	VERSION INTERFACES .....	25
3.5.1	<i>Rtp_GetVersion</i> .....	25

## List of Tables

TABLE 2-1	BASIC DATA TYPES .....	5
TABLE 2-2	PLATFORM TYPES .....	6
TABLE 3-1	EN_RTP_PAYLOAD_TYPE .....	8
TABLE 3-2	ST_RTP_RTP_INFO .....	9
TABLE 3-3	ST_RTP_RTCP_INFO .....	9
TABLE 3-4	ST_RTP_USER_INFO .....	10
TABLE 3-5	PFN RTPDATAIND .....	10
TABLE 3-6	PFN RTCPDATAIND .....	11
TABLE 3-7	ST_RTP_RTP_MSG .....	11
TABLE 3-8	ST_RTP_CFG .....	13
TABLE 3-9	ST_ZOS_INET_ADDR .....	14
TABLE 3-10	ST_ZOS_INET_IP .....	15

## List of Figures

FIGURE 2-1	RTP STACK FRAMEWORK .....	5
------------	---------------------------	---

## 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.

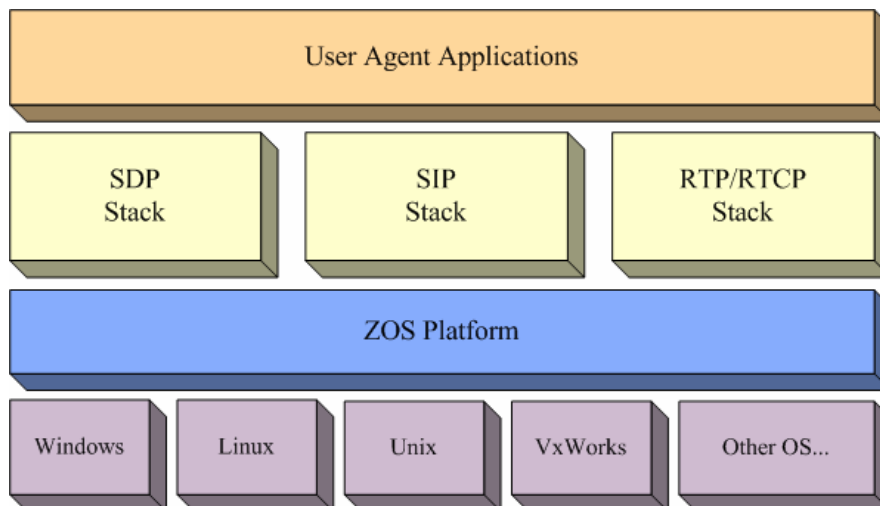


Figure 2-1 RTP Stack Framework

### 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

Here are the interfaces of protocol RTP provided to users. They are listed as the followings:

### 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,          /* pcmu (g711u) */
    EN_RTP_PAYLOAD_GSM = 3,          /* gsm */
    EN_RTP_PAYLOAD_G723 = 4,        /* g723 */
    EN_RTP_PAYLOAD_PCMA = 8,        /* pcma (g711a) */
    EN_RTP_PAYLOAD_G722 = 9,        /* g722 */
    EN_RTP_PAYLOAD_G728 = 15,       /* g728 */
    EN_RTP_PAYLOAD_G729 = 18,       /* g729 */
    EN_RTP_PAYLOAD_JPEG = 26,       /* jpeg */
    EN_RTP_PAYLOAD_H261 = 31,       /* h261 */
    EN_RTP_PAYLOAD_H263 = 34,       /* h263 */
    EN_RTP_PAYLOAD_MAX = 128        /* max payload types */
} 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_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
{
    UCHAR ucVer;                /* rtp version */
    UCHAR ucPadding;           /* padding flag */
    UCHAR ucExt;               /* extension exist flag */
    UCHAR ucCsrcCount;        /* contributing source count */
    UCHAR ucMarker;           /* marker flag */
    UCHAR ucPayload;          /* payload type */
    USHORT 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\_RTP\_USER\_INFO

It is the user information structure defined in rtp\_type.h.

```
typedef struct tagRTP_USER_INFO
{
    ZTASKID zTaskId;           /* user task ID */
    ZULONG dwUserId;          /* user ID */
    PFN_RTPMALLOC pfnMalloc;   /* data malloc */
    PFN_RTPFREE pfnFree;       /* data free */
    PFN\_RTPDATAIND pfnDataInd; /* data indication */
} ST_RTP_USER_INFO;
```

Its members are described below:

Member	Description
zTaskId	ID of a task which RTP stack will inform of receiving a valid RTP packet by sending a ZOS message. There are two alternatives provided to users when RTP stack receives a valid RTP packet. One is that, by assigning ZMAXTASKID to zTaskId, RTP stack calls user-defined pfnMalloc to allocate

	memory for the RTP packet, then to process the packet and calls user-defined pfnFree to free the memory finally. Another one is that, by assigning a valid value, except ZMAXTASKID, to zTaskId, RTP stack uses ZOS function to allocate and free memory for the RTP packet, and inform the task, specified by zTaskId, with a ZOS task message containing the RTP packet.
dwUserId	User ID. It is used to identify the user of RTP stack. It is saved in RTP stack when Rtp_UiOpenReq is invoked and returned to the user when the stack informs the user of receiving an RTP packet. The stack itself does not limit on the allowable values of dwUserId.
pfnMalloc	User-defined memory allocation function. It must have a prototype of PFN_RTPMALLOC. Refer to the description of <a href="#">zTaskId</a> .
pfnFree	User-defined memory free function. It must have a prototype of PFN_RTPFREE. Refer to the description of <a href="#">zTaskId</a> .
pfnDataInd	User-defined RTP packet process function. It must have a prototype of <a href="#">PFN RTPDATAIND</a> . Refer to the description of <a href="#">zTaskId</a> .

Table 3-4 ST\_RTP\_USER\_INFO

### 3.1.5 PFN\_RTPDATAIND

It is a function provided by users to RTP stack which calls it to process received RTP packets.

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

Its parameters are described below:

Parameter	Description
dwUserId	User ID. It identifies the users of RTP stack
dwRtpId	The RTP id from which RTP packets are received.
pstRmtAddr	The remote address from which an RTP packet is received.
pcMem	The memory which store RTP packets
pcData	A pointer to the data included in an RTP packet.
iDataLen	Length of the RTP data included in an RTP packet.
pstInfo	RTP information

Table 3-5 PFN\_RTPDATAIND

### 3.1.6 PFN\_RTCPAPPIND

It is a function provided by users to RTP stack which calls it to process received RTCP app packets.

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

Its parameters are described below:

Parameter	Description
dwUserId	User ID. It identifies the users of RTP stack
dwRtpId	The RTP id from which RTP packets are received.
pstInfo	RTCP information.
pucAppData	The data of application.
iAppLen	A pointer to the application data.

Table 3-6 PFN\_RTCPDATAIND

### 3.1.7 ST\_RTP\_RTP\_MSG

It is the inform message structure, included in a ZOS task message structure, which is used by RTP stack to inform a task of receiving an RTP packet if the task ID is not ZMAXTASKID. It includes a pointer pstMsgBuf to ZOS data buffer structure where the received RTP packet is stored. Please refer to the description of [zTaskId](#).

```

/* RTP indication message (use ZOS message indication) */
typedef struct tagRTP_RTP_MSG
{
    ZULONG dwUserId;           /* user id */
    ZULONG dwRtpId;           /* RTP id */
    ZULONG dwSeqNum;          /* sequence number */
    ST_ZOS_INET_ADDR stRmtAddr; /* remote address */
    ST_ZOS_DBUF *pstMsgBuf;    /* message dbuf */
} ST_RTP_RTP_MSG;

```

Its members are described below:

Member	Description
dwUserId	User ID. It represents the user who should receive the RTP packet.
wRtpPort	The local RTP port from which the RTP packet is received.
wSeqNum	Sequence number in the header of the received RTP packet.
pstRmtAddr	The remote address from which the RTP packet is received.
pstMsgBuf	A pointer to ZOS data buffer structure where the received RTP packet is stored.

Table 3-7 ST\_RTP\_RTP\_MSG

### 3.1.8 ST\_RTP\_CFG

It is the RTP config structure defined in rtp\_cfg.h.

```

typedef struct tagRTP_CFG
{
    /* log config */
    ZCHAR *pcLogFileName;           /* log file name */
    ZULONG dwLogOpt;                /* log option */
    ZULONG dwLogLevel;              /* log level */
    ZULONG dwLogBufSize;            /* log buffer size */

    /* session task config */
    ZINT iSessTaskPriority;          /* session task priority */
    ZULONG dwSessTaskStackSize;     /* session task stack size */
    ZULONG dwSessTaskQueueSize;     /* session task queue size */
    ZULONG dwSessTaskTimerNum;      /* session task timer number */

    /* transport task config */
    ZINT iTptTaskPriority;           /* transport task priority */
    ZULONG dwTptTaskQueueSize;      /* transport task queue size */
    ZINT iTptSelectTimeOut;         /* transport select timeout */

    /* session config */
    ZINT iPtptMaxNum;               /* participant maximum number */
    ZINT iConnMaxNum;               /* transport connection maximum number */
    ZINT iSenderMaxNum;              /* sender max number */

    /* local address config */
    ST_ZOS_INET_IP stLocalIp;       /* local ip address */
    ZULONG dwPortBegin;             /* RTP port begin */

    /* RTP property config */
    ZUINT iRtcpBandwidth;            /* RTCP bandwidth */
    ZUINT iSenderLeastShare;         /* sender's least share in bandwidth */
    ZUINT iRecverMostShare;          /* receiver's most share in bandwidth */
} ST_RTP_CFG;

```

Its members are described below:

Member	Description
pcLogFileName	A character pointer to a string which represents a log file name.
dwLogOpt	A log option. Allowable values: ZLOG_OPT_NULL: No option. ZLOG_OPT_MUTEX: Asynchronism. ZLOG_OPT_PRINT: Print the log information.

dwLogLevel	<p>Represents the log level. Its value can be <b>ZLOG_LEVEL_NULL</b>, or <b>ZLOG_LEVEL_ALL</b>, or combination of one or more than one of <b>ZLOG_LEVEL_FATAL</b>, <b>ZLOG_LEVEL_ERROR</b>, <b>ZLOG_LEVEL_WARNING</b>, <b>ZLOG_LEVEL_INFO</b>, and <b>ZLOG_LEVEL_DBG</b>.</p> <p><b>ZLOG_LEVEL_NULL</b>: Log nothing.</p> <p><b>ZLOG_LEVEL_FATAL</b>: Log fatal information.</p> <p><b>ZLOG_LEVEL_ERROR</b>: Log error information.</p> <p><b>ZLOG_LEVEL_WARNING</b>: Log warning information.</p> <p><b>ZLOG_LEVEL_INFO</b>: Log normal information.</p> <p><b>ZLOG_LEVEL_DBG</b>: Log debug information.</p> <p><b>ZLOG_LEVEL_ALL</b>: Log all information.</p>
dwLogBufSize	Size of the log buffer.
iSessTaskPriority	<p>Session task priority.</p> <p>Allowable values:</p> <p><b>ZTASK_PRIORITY_MAX</b>: Maximum priority.</p> <p><b>ZTASK_PRIORITY_NORMAL</b>: Normal priority.</p> <p><b>ZTASK_PRIORITY_MIN</b>: Minimum priority.</p> <p><b>ZTASK_PRIORITY_INCREMENT</b>: Increasable priority.</p>
dwSessTaskStackSize	Size of session task stack.
dwSessTaskQueueSize	Size of session task queue.
dwSessTaskTimerNum	Number of session task timers.
iTptTaskPriority	<p>Transport task priority.</p> <p>Allowable values:</p> <p><b>ZTASK_PRIORITY_MAX</b>: Maximum priority.</p> <p><b>ZTASK_PRIORITY_NORMAL</b>: Normal priority.</p> <p><b>ZTASK_PRIORITY_MIN</b>: Minimum priority.</p> <p><b>ZTASK_PRIORITY_INCREMENT</b>: Increasable priority.</p>
dwTptTaskQueueSize	Size of transport task queue.
iTptSelectTimeOut	Specifies when time is out. Function <i>Zos_SocketSelect()</i> pends on a set of file descriptors. If there is any readable or writable socket file descriptor, it will return ZOK or wait until time is out.
iPptMaxNum	The maximum number of session participants.
iConnMaxNum	The maximum number of transport connections.
iSenderMaxNum	The maximum number of senders.
dwLocalIp	Local IP address.
dwRtpPortBegin	The number of first allocable RTP port. Its default value is 37000.
iRtcpBandwidth	The RTCP bandwidth used by RTP stack
iSenderLeastShare	The least proportion of total RTCP bandwidth occupied by the senders in an RTP session.
iRecverMostShare	The most proportion of total RTCP bandwidth occupied by the receivers in an RTP session.

Table 3-8 ST\_RTP\_CFG

### 3.1.9 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
<code>ucType</code>	Network address type, Its value can be <code>ZINET_IPV4</code> or <code>ZINET_IPV6</code> .
<code>wPort</code>	A port number which is order independent.
<code>u.dwIp</code>	An IPv4 address which is order independent.
<code>u.aucIp</code>	An array used to store the IPv4 address ( <i>u.dwIp</i> , consisting of 4 bytes), with each element holding one byte of the IPv4 address.
<code>u.aucIpv6</code>	An array used to store the IPv6 address.

Table 3-9 ST\_ZOS\_INET\_ADDR

### 3.1.10 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.dwlp	An IPv4 address which is order independent.
u.auclp	An array used to store the IPv4 address ( <i>u.dwlp</i> , consisting of 4 bytes), with each element holding one byte of the IPv4 address.
u. auclpv6	An array used to store the IPv6 address.

Table 3-10 ST\_ZOS\_INET\_IP

### 3.2 Config Interfaces

#### 3.2.1 Rtp\_CfgInit

This function is invoked to give RTP stack default configuration parameters.

```
ZINT Rtp_CfgInit();
```

[Parameters]

**Input parameter:**

None.

**Output parameter:**

None.

[Return value]

```
ZOK: operation succeeded
ZFAILED: operation failed
```

[Requirements]

Routine	Required header	Library
Rtp_CfgInit	rtp.h	rtp.lib

### 3.3 Task Interfaces

#### 3.3.1 Rtp\_Start

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

```
ZINT Rtp_Start();
```

[Parameters]

**Input parameter:**

None.

**Output parameter:**

None.

**[Return value]**

ZOK: operation succeeded  
 ZFAILED: operation failed

**[Requirements]**

Routine	Required header	Library
Rtp_Start	rtp.h	rtp.lib

**3.4 Upper Interfaces**

Here are upper interfaces provided to users.

**3.4.1 Rtp\_Open**

This function is used to open an RTP channel.

```
ZINT Rtp_Open(ST\_RTP\_USER\_INFO *pstUserInfo, ZUCHAR ucPayload,
              ZUSHORT *pwRtpPort, ZULONG *pdwRtpId);
```

**[Parameters]****Input parameter:**

[ST\\_RTP\\_USER\\_INFO](#) \*pstUserInfo

A pointer to user information structure where user information is stored. If NULL, ZFAILED will be returned.

ZUCHAR ucPayload

Indicates the payload type. For detailed information please refer to [EN\\_RTP\\_PAYLOAD\\_TYPE](#).

**Output parameter:**

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]**

ZOK: operation succeeded  
 ZFAILED: operation failed

**[Requirements]**

Routine	Required header	Library
Rtp_Open	rtp.h	rtp.lib

### 3.4.2 Rtp\_OpenX

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

```
ZINT Rtp_Open(ST\_RTP\_USER\_INFO *pstUserInfo, ZUCHAR ucPayload,
              ZUSHORT wRtpPort, ZULONG *pdwRtpId);
```

[Parameters]

**Input parameter:**

[ST\\_RTP\\_USER\\_INFO](#) \*pstUserInfo

A pointer to user information structure where user information is stored. If NULL, ZFAILED will be returned.

ZUCHAR ucPayload

Indicates the payload type. For detailed information please refer to [EN\\_RTP\\_PAYLOAD\\_TYPE](#).

ZUSHORT wRtpPort

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

**Output parameter:**

ZULONG \*pdwRtpId

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

[Return value]

ZOK: operation succeeded

ZFAILED: operation failed

[Requirements]

Routine	Required header	Library
Rtp_OpenX	rtp.h	rtp.lib

### 3.4.3 Rtp\_Close

This function is used to close an RTP channel.

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

[Parameters]

**Input parameter:**

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 parameter:**

None.

[Return value]

ZOK: operation succeeded  
ZFAILED: operation failed

[Requirements]

Routine	Required header	Library
Rtp_Close	rtp.h	rtp.lib

### 3.4.4 Rtp\_RtcpOpen

This function is used to open an RTCP channel.

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

[Parameters]

**Input parameter:**

ZULONG *dwRtpId*  
A RTP channel ID.

**Output parameter:**

ZUSHORT *\*pwRtcpPort*  
A pointer to the opened RTCP port as the result of opening a channel.

[Return value]

ZOK: operation succeeded  
ZFAILED: operation failed

[Requirements]

Routine	Required header	Library
Rtp_RtcpOpen	rtp.h	rtp.lib

### 3.4.5 Rtp\_RtcpOpenX

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

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

[Parameters]

**Input parameter:**

ZULONG *dwRtpId*

A RTP channel ID.

ZUSHORT *wRtcpPort*

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

**Output parameter:**

None.

[Return value]

ZOK: operation succeeded

ZFAILED: operation failed

[Requirements]

Routine	Required header	Library
Rtp_RtcpOpenX	rtp.h	rtp.lib

### 3.4.6 Rtp\_RtcpEnable

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

ZINT Rtp\_RtcpEnable(ZULONG *dwRtpId*, ZBOOL *bEnable*)

[Parameters]

**Input parameter:**

ZULONG *dwRtpId*

A RTP channel ID.

ZBOOL *bEnable*

A Boolean value indicates to enable or disable RTCP.

**Output parameter:**

None.

[Return value]

ZOK: operation succeeded

ZFAILED: operation failed

[Requirements]

Routine	Required header	Library
Rtp_RtcpEnable	rtp.h	rtp.lib

### 3.4.7 Rtp\_RtpSend

This function is used to send RTP data.

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

## [Parameters]

**Input parameter:**

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 parameter:**

None.

## [Return value]

ZOK: operation succeeded

ZFAILED: operation failed

## [Requirements]

Routine	Required header	Library
Rtp_RtpSend	rtp.h	rtp.lib

### 3.4.8 Rtp\_RtpSendM

This function is used to send RTP data with mark.

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

## [Parameters]

**Input parameter:**

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 parameter:**

None.

## [Return value]

ZOK: operation succeeded  
 ZFAILED: operation failed

## [Requirements]

Routine	Required header	Library
Rtp_RtpSendM	rtp.h	rtp.lib

### 3.4.9 Rtp\_RtpResend

This function is used to resend RTP data.

```
ZINT Rtp_RtpResend(ZULONG dwRtpId);
```

## [Parameters]

**Input parameter:**

ZULONG *dwRtpId*

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

**Output parameter:**

None.

## [Return value]

ZOK: operation succeeded  
 ZFAILED: operation failed

## [Requirements]

Routine	Required header	Library
Rtp_RtpResend	rtp.h	rtp.lib

### 3.4.10 Rtp\_RtcpAppSend

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

```
ZINT Rtp_RtcpAppSend(ZULONG dwRtpId, ZUCHAR ucSubType,  

  ZUCHAR *pucData, ZULONG dwLen);
```

## [Parameters]

**Input parameter:**

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.

ZULONG *dwLen*

Length of RTCP APP data.

#### Output parameter:

None.

#### [Return value]

ZOK: operation succeeded

ZFAILED: operation failed

#### [Requirements]

Routine	Required header	Library
Rtp_RtcpAppSend	rtp.h	rtp.lib

### 3.4.11 Rtp\_RtcpAppResend

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

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

#### [Parameters]

##### Input parameter:

ZULONG *dwRtpId*

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

#### Output parameter:

None.

#### [Return value]

ZOK: operation succeeded

ZFAILED: operation failed

#### [Requirements]

Routine	Required header	Library
Rtp_RtcpAppResend	rtp.h	rtp.lib

### 3.4.12 Rtp\_SetRmtAddr

This function is used to set the remote address.

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

#### [Parameters]

##### Input parameter:

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 parameter:**

None.

[Return value]

ZOK: operation succeeded

ZFAILED: operation failed

[Requirements]

Routine	Required header	Library
Rtp_SetRmtAddr	rtp.h	rtp.lib

### 3.4.13 Rtp\_SetPayload

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

ZINT Rtp\_SetPayload(ZULONG *dwRtpId*, ZUCHAR *ucPayload*);

[Parameters]

**Input parameter:**

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 parameter:**

None.

[Return value]

ZOK: operation succeeded

ZFAILED: operation failed

[Requirements]

Routine	Required header	Library
Rtp_SetPayload	rtp.h	rtp.lib

### 3.4.14 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 parameter:**

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 extension bit.

ZUINT *iTsInc*

Indicates the timestamp increase.

**Output parameter:**

None.

[Return value]

ZOK: operation succeeded

ZFAILED: operation failed

[Requirements]

Routine	Required header	Library
Rtp_SetProf	rtp.h	rtp.lib

### 3.4.15 Rtp\_SetRtcpInd

This function is used to set the process function while receiving RTCP APP packet.

```
ZINT Rtp_SetRtcpInd(ZULONG dwRtpId, PFN\_RTCPAPPIND pfn_RtcpAppInd)
```

[Parameters]

**Input parameter:**

ZULONG *dwRtpId*  
 The payload profile information.

[PFN\\_RTCPAPPIND](#) *pfn\_RtcpAppInd*  
 Indicates process function while receiving RTCP APP packet.

**Output parameter:**

None.

[Return value]

ZOK: operation succeeded  
 ZFAILED: operation failed

[Requirements]

Routine	Required header	Library
Rtp_SetRtcpInd	rtp.h	rtp.lib

### 3.4.16 Rtp\_GetSsrc

This function is used to get the .

ZUINT Rtp\_GetSsrc(ZULONG *dwRtpId*);

[Parameters]

**Input parameter:**

ZULONG *dwRtpId*  
 The payload profile information.

**Output parameter:**

None.

[Return value]

The synchronization source of the RTP channel.

[Requirements]

Routine	Required header	Library
Rtp_GetSsrc	rtp.h	rtp.lib

## 3.5 Version Interfaces

### 3.5.1 Rtp\_GetVersion

This function is used to get the RTP version.

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

[Parameters]

**Input parameter:**

None.

**Output parameter:**

None.

[Return value]

ZOK: operation succeeded

ZFAILED: operation failed

[Requirements]

Routine	Required header	Library
Rtp_GetVersion	rtp.h	rtp.lib