

---

Juphoon Protocol Framework

# Session Initiation Protocol

Published: Dec 2009

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

---

## Juphoon SIP Function Definition

Juphoon System Software Corporation.

<http://www.juphoon.com>

Tel: +86-574-87287820

Fax: +86-574-87304379

Text Part Number: 101-004-01-01

Copyright © 2009, Juphoon System Software Corporation.

All rights reserved.

## Contents

SESSION INITIATION PROTOCOL .....	1
JUPHOON SIP FUNCTION DEFINITION .....	1
<b>1. INTRODUCTION.....</b>	<b>8</b>
1.1    PURPOSE.....	8
1.2    AUDIENCE .....	8
1.3    SCOPE.....	8
1.4    DEFINITIONS, ACRONYMS, AND ABBREVIATIONS.....	8
<b>2. SYSTEM ENVIRONMENT.....</b>	<b>9</b>
2.1    BASIC DATA TYPES .....	9
2.2    PLATFORM TYPES.....	10
2.3    COMMON CONCEPTS .....	10
2.3.1  Module ID.....	10
2.3.2  Instance ID .....	11
2.3.3  Task ID .....	11
2.3.4  Protocol ID.....	11
2.4    SIP PRIMITIVES .....	11
<b>3. USER INTERFACES .....</b>	<b>14</b>
3.1    INTERFACE STRUCTURES .....	14
3.1.1  ZOS Structure .....	14
3.1.2  SIP Message Structures.....	16
3.1.3  SIP Headers and Associated Sub-Structures.....	35
3.1.4  SIP Configuration Interfaces.....	38
3.2    ABNF INTERFACES .....	39
3.2.1  Sip_AbnfInit.....	39
3.2.2  Sip_AbnfDestroy.....	40
3.2.3  Sip_DecodeMsg.....	40
3.2.4  Sip_EncodeMsg .....	41
3.2.5  Sip_CreateMsgHdr.....	41
3.2.6  Sip_DeleteMsgHdr .....	42
3.2.7  Sip_FillMsgHdr.....	42
3.2.8  Sip_FillExtHdr.....	43
3.2.9  Sip_FindMsgHdr .....	43
3.2.10  Sip_FindMsgHdrX.....	44
3.2.11  Sip_FindExtHdr.....	44
3.2.12  Sip_FindExtHdrX.....	45
3.2.13  Sip_GetMimeBoundary .....	45
3.2.14  Sip_GetMimeBody.....	46
3.2.15  Sip_GetContentLen.....	46
3.2.16  Sip_GetContentLen2.....	47

3.2.17	<i>Sip_UpdateContentLen</i> .....	47
3.2.18	<i>Sip_GetViaBranch</i> .....	48
3.2.19	<i>Sip_HdrLstCreateHdr</i> .....	48
3.2.20	<i>Sip_HdrLstDeleteHdr</i> .....	49
3.2.21	<i>Sip_HdrLstFindHdr</i> .....	49
3.2.22	<i>Sip_HdrLstGetContentLen</i> .....	50
3.2.23	<i>Sip_HdrLstUpdateContentLen</i> .....	50
3.2.24	<i>Sip_MPartLstCreateMPart</i> .....	51
3.2.25	<i>Sip_AbnfGetVersion</i> .....	51
3.3	ABNF CONFIG INTERFACES .....	52
3.3.1	<i>Sip_AbnfCfgGetLogLevel</i> .....	52
3.3.2	<i>Sip_AbnfCfgGetOption</i> .....	52
3.3.3	<i>Sip_AbnfCfgGetHdrDecode</i> .....	52
3.3.4	<i>Sip_AbnfCfgSetLogLevel</i> .....	53
3.3.5	<i>Sip_AbnfCfgSetOption</i> .....	53
3.3.6	<i>Sip_AbnfCfgClrOption</i> .....	54
3.3.7	<i>Sip_AbnfCfgSetHdrDecode</i> .....	54
3.4	PRIMITIVE INTERFACES .....	55
3.4.1	<i>Sip_SendSimReq</i> .....	55
3.4.2	<i>Sip_SendSimRsp</i> .....	55
3.4.3	<i>Sip_SendSsmReq</i> .....	56
3.4.4	<i>Sip_SendSsmRsp</i> .....	56
3.4.5	<i>Sip_SendSamReq</i> .....	57
3.4.6	<i>Sip_SendScmReq</i> .....	58
3.4.7	<i>Sip_SendSmmReq</i> .....	58
3.4.8	<i>Sip_SendSmmRsp</i> .....	59
3.4.9	<i>Sip_SendStmReq</i> .....	60
3.4.10	<i>Sip_SendDamReq</i> .....	60
3.4.11	<i>Sip_SendDamReqX</i> .....	61
3.4.12	<i>Sip_SendDamRsp</i> .....	62
3.4.13	<i>Sip_SendCimReq</i> .....	62
3.4.14	<i>Sip_SendCimRsp</i> .....	63
3.4.15	<i>Indication and Confirmation Messages</i> .....	64
3.5	MESSAGE INTERFACES .....	65
3.5.1	<i>Sip_MsgCreate</i> .....	65
3.5.2	<i>Sip_MsgCreateX</i> .....	66
3.5.3	<i>Sip_MsgDelete</i> .....	66
3.5.4	<i>Sip_MsgFillReqLineByIp</i> .....	67
3.5.5	<i>Sip_MsgFillReqLineByName</i> .....	67
3.5.6	<i>Sip_MsgFillReqLineBySipUri</i> .....	68
3.5.7	<i>Sip_MsgFillReqLineByTelUri</i> .....	69
3.5.8	<i>Sip_MsgFillStatusLine</i> .....	70
3.5.9	<i>Sip_MsgFillHdrCallId</i> .....	70
3.5.10	<i>Sip_MsgFillHdrCSeq</i> .....	71

3.5.11	<i>Sip_MsgFillHdrSupted</i> .....	71
3.5.12	<i>Sip_MsgFillHdrAllow</i> .....	72
3.5.13	<i>Sip_MsgFillHdrRequire</i> .....	72
3.5.14	<i>Sip_MsgFillHdrSessExpire</i> .....	73
3.5.15	<i>Sip_MsgFillHdrMinSe</i> .....	73
3.5.16	<i>Sip_MsgFillHdrEvt</i> .....	74
3.5.17	<i>Sip_MsgFillHdrExpire</i> .....	74
3.5.18	<i>Sip_MsgFillHdrSubsSta</i> .....	75
3.5.19	<i>Sip_MsgFillHdrRAck</i> .....	75
3.5.20	<i>Sip_MsgFillHdrPrivacy</i> .....	76
3.5.21	<i>Sip_MsgGetFromToTag</i> .....	76
3.5.22	<i>Sip_HdrFillCseq</i> .....	77
3.5.23	<i>Sip_HdrFillFromToByIp</i> .....	77
3.5.24	<i>Sip_HdrFillFromToByName</i> .....	78
3.5.25	<i>Sip_HdrFillFromToBySipUri</i> .....	79
3.5.26	<i>Sip_HdrFillFromToBySipsUri</i> .....	80
3.5.27	<i>Sip_HdrFillFromToByTelUri</i> .....	81
3.5.28	<i>Sip_HdrFillFromToByImUri</i> .....	81
3.5.29	<i>Sip_HdrFillReferToByIp</i> .....	82
3.5.30	<i>Sip_HdrFillReferToByName</i> .....	83
3.5.31	<i>Sip_HdrFillReferredByByIp</i> .....	83
3.5.32	<i>Sip_HdrFillReferredByByName</i> .....	84
3.5.33	<i>Sip_HdrFillUserAgent</i> .....	85
3.5.34	<i>Sip_HdrFillServer</i> .....	86
3.5.35	<i>Sip_HdrFillEvt</i> .....	87
3.5.36	<i>Sip_HdrFillSubsSta</i> .....	87
3.5.37	<i>Sip_HdrFillRAck</i> .....	88
3.5.38	<i>Sip_HdrFillPrivacy</i> .....	88
3.5.39	<i>Sip_HdrFromToAddTag</i> .....	89
3.5.40	<i>Sip_HdrAllowAddMethod</i> .....	89
3.5.41	<i>Sip_HdrSuptedAddTag</i> .....	90
3.5.42	<i>Sip_HdrSuptedAddTagX</i> .....	91
3.5.43	<i>Sip_HdrRequireAddTag</i> .....	92
3.5.44	<i>Sip_HdrRequireAddTagX</i> .....	93
3.5.45	<i>Sip_HdrReferToAddHdr</i> .....	93
3.5.46	<i>Sip_HdrEvtAddId</i> .....	94
3.5.47	<i>Sip_HdrSubsStaAddReasonVal</i> .....	95
3.5.48	<i>Sip_HdrSubsStaAddExpires</i> .....	95
3.5.49	<i>Sip_HdrSubsStaAddRetryAfter</i> .....	96
3.5.50	<i>Sip_HdrReplaceAddFromTag</i> .....	97
3.5.51	<i>Sip_HdrReplaceAddToTag</i> .....	97
3.5.52	<i>Sip_HdrReplaceAddEarlyTag</i> .....	98
3.5.53	<i>Sip_HdrFromToGetTag</i> .....	98
3.5.54	<i>Sip_ParmFillStatusLine</i> .....	99

3.5.55	<i>Sip_ParmFillSipUri</i> .....	99
3.5.56	<i>Sip_ParmFillTelUri</i> .....	100
3.5.57	<i>Sip_ParmFillImUri</i> .....	100
3.5.58	<i>Sip_ParmFillReqUriByIp</i> .....	101
3.5.59	<i>Sip_ParmFillReqUriByName</i> .....	101
3.5.60	<i>Sip_ParmFillAddrSpecByIp</i> .....	102
3.5.61	<i>Sip_ParmFillAddrSpecByName</i> .....	103
3.5.62	<i>Sip_ParmFillAddrSpecBySipUri</i> .....	104
3.5.63	<i>Sip_ParmFillAddrSpecBySipsUri</i> .....	104
3.5.64	<i>Sip_ParmFillAddrSpecByTelUri</i> .....	105
3.5.65	<i>Sip_ParmFillAddrSpecByImUri</i> .....	105
3.5.66	<i>Sip_ParmFillContactParm</i> .....	106
3.5.67	<i>Sip_ParmFillHostPort</i> .....	106
3.5.68	<i>Sip_ParmFillViaSentProtocol</i> .....	107
3.5.69	<i>Sip_ParmFillViaSentBy</i> .....	107
3.5.70	<i>Sip_ParmFillViaBranch</i> .....	108
3.5.71	<i>Sip_ParmFillViaRecv</i> .....	108
3.5.72	<i>Sip_ParmFillViaRport</i> .....	109
3.5.73	<i>Sip_ParmFillEvtType</i> .....	109
3.5.74	<i>Sip_ParmFillEvtPkg</i> .....	110
3.5.75	<i>Sip_ParmFillEvtPkgX</i> .....	111
3.5.76	<i>Sip_ParmFillEvtTemp</i> .....	111
3.5.77	<i>Sip_ParmFillDispName</i> .....	112
3.5.78	<i>Sip_ParmFillCredents</i> .....	112
3.5.79	<i>Sip_ParmFillDRspUserName</i> .....	113
3.5.80	<i>Sip_ParmFillDRspRealm</i> .....	114
3.5.81	<i>Sip_ParmFillDRspNonce</i> .....	114
3.5.82	<i>Sip_ParmFillDRspUri</i> .....	115
3.5.83	<i>Sip_ParmFillDRspRsp</i> .....	115
3.5.84	<i>Sip_ParmFillDRspOpaque</i> .....	116
3.5.85	<i>Sip_ParmFillDRspAlgo</i> .....	116
3.5.86	<i>Sip_ParmFillContactIsFocus</i> .....	117
3.5.87	<i>Sip_ParmFillContactPocTb</i> .....	117
3.5.88	<i>Sip_ParmFillContact3gppCv</i> .....	118
3.5.89	<i>Sip_ParmFillAcValPocTb</i> .....	118
3.5.90	<i>Sip_ParmFillAcVal3gppCv</i> .....	119
3.5.91	<i>Sip_ParmFillAcValRequire</i> .....	119
3.5.92	<i>Sip_ParmFillAcValExplict</i> .....	120
3.5.93	<i>Sip_ParmFillMediaType</i> .....	120
3.5.94	<i>Sip_ParmOptTagLstAddTag</i> .....	121
3.5.95	<i>Sip_ParmOptTagLstGetTag</i> .....	121
3.5.96	<i>Sip_ParmFromToLstAddTag</i> .....	122
3.5.97	<i>Sip_ParmFromToLstGetTag</i> .....	122
3.5.98	<i>Sip_ParmEvtLstGetId</i> .....	123

3.5.99	<i>Sip_ParmContactLstGetExpire</i> .....	123
3.5.100	<i>Sip_ParmMethodLstGetMethod</i> .....	124
3.5.101	<i>Sip_ParmMediaLstAddAttr</i> .....	124
3.6	UTILITY INTERFACES .....	125
3.6.1	<i>Sip_LogOpen</i> .....	125
3.6.2	<i>Sip_LogClose</i> .....	126
3.6.3	<i>Sip_SessEvtFree</i> .....	126
3.6.4	<i>Sip_ReasonFromCode</i> .....	126
3.6.5	<i>Sip_GetMethodDesc</i> .....	127
3.6.6	<i>Sip_GetSessEvtDesc</i> .....	127
3.6.7	<i>Sip_GetStatCodeDesc</i> .....	128
3.7	TASK INTERFACES .....	128
3.7.1	<i>Sip_Start</i> .....	128
3.7.2	<i>Sip_Stop</i> .....	128
3.7.3	<i>Sip_Restart</i> .....	129
3.8	VERSION INTERFACES .....	129
3.8.1	<i>Sip_GetVersion</i> .....	129
<b>4.</b>	<b>INTERFACE PROCEDURES .....</b>	<b>131</b>
4.1	NORMAL SIP PROCEDURE .....	131
4.2	SUCCESSFUL OUTGOING CALL SETUP .....	131
4.3	SUCCESSFUL INCOMING SIP CALL .....	132
4.4	SESSION STATUS REQUEST .....	133
4.5	SESSION STATUS RESPONSE .....	134
4.6	RE-INVITE REQUEST .....	134
4.7	RESPONSE TO RE-INVITE REQUEST .....	134
4.8	BYE REQUEST .....	135
4.9	RESPONSE TO BYE REQUEST .....	135
<b>5.</b>	<b>INTERFACE EXAMPLES .....</b>	<b>137</b>
5.1	TEST SIP STRUCTURES .....	137
5.2	TEST SIP MACROS .....	137
5.3	TEST SIP INTERFACES .....	138
5.4	<i>SIP_DECODEMSG</i> .....	142
5.5	<i>SIP_ENCODEMSG</i> .....	144
5.6	<i>SIP_SENDSIMREQ</i> .....	144
5.7	<i>SIP_SENDSIMRSP</i> .....	146
5.8	<i>SIP_SENDSMREQ</i> .....	146
5.9	<i>SIP_SENDSAMREQ</i> .....	148
5.10	<i>SIP_SENDSAMREQ</i> .....	148
5.11	<i>SIP_SENDSMMREQ</i> .....	149
5.12	<i>SIP_SENDSMMRSP</i> .....	149
5.13	<i>SIP_SENSTMREQ</i> .....	150
5.14	<i>SIP_SENDDAMREQ</i> .....	151
5.15	<i>SIP_SENDCIMREQ</i> .....	152

5.16	SIP_SENDCIMRSP.....	153
------	---------------------	-----

## **List of Tables**

Table 2-1	Basic Data Types .....	10
Table 2-2	Platform Types.....	10
Table 3-1	ST_ABNF_MSG .....	15
Table 3-2	ST_ZOS_MSP .....	16
Table 3-3	ST_SIP_MSG .....	17
Table 3-4	ST_SIP_SESS_EVNT .....	19
Table 3-5	EN_SIP_SESS_EVNT_TYPE.....	22
Table 3-6	Headers and Associated Sub-Structure.....	38
Table 3-7	Properties Description.....	39

## **List of Figures**

Figure 2-1	SIP Stack Framework.....	9
Figure 2-2	Use Primitives to Send Requests and Responses .....	13
Figure 4-1	A Normal SIP Procedure between Two User Agents.....	131
Figure 4-2	Successful Outgoing SIP Call Setup .....	132
Figure 4-3	Successful Incoming SIP Call Setup .....	133
Figure 4-4	Session Status Request .....	133
Figure 4-5	Session Status Response.....	134
Figure 4-6	A re-INVITE Request .....	134
Figure 4-7	The Reponse to a re-INVITE Request.....	135
Figure 4-8	A BYE Request .....	135
Figure 4-9	The Reponse to a BYE Request .....	136

## 1. Introduction

Session Initiation Protocol (SIP) is an application-layer control (signaling) protocol for creating, modifying, and terminating sessions with one or more participants. These sessions include Internet telephone calls, multimedia distribution, and multimedia conferences.

SIP invitations used to create sessions carrying session descriptions that allow participants to agree on a set of compatible media types. SIP makes use of elements called proxy servers to help route requests to the user's current location, authenticate and authorize users for services, implement provider call-routing policies, and provide features to users. SIP also provides a registration function that allows users to upload their current locations for use by proxy servers. SIP runs on top of several different transport protocols.

### 1.1 Purpose

This document is provided to developers who will develop their own products with the functions of SIP furnished by Juphoon.

### 1.2 Audience

The readers of this document are assumed to have a working knowledge of SIP.

### 1.3 Scope

This document provides several functions of SIP and their usages but not the description about the design and the realization of SIP.

### 1.4 Definitions, Acronyms, and Abbreviations

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

Abbreviation	Description
ABNF	Augmented BNF
IM	Instant Message
MTU	Maximum Transmission Unit
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SUA	SIP User Agent
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
ZOS	Zero Operating System

## 2. System Environment

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

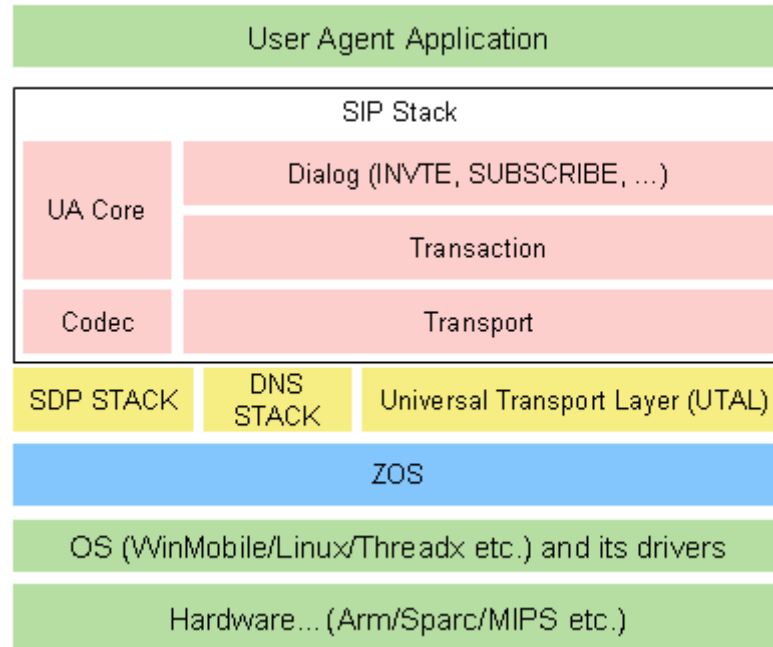


Figure 2-1 SIP Stack Framework

### 2.1 Basic Data Types

There are some basic data types provided by ZOS platform. Table 2-1 lists these types used by the SIP Stack and user agent applications.

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 2-3 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
ZEVNTID	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 SIP 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 module, the ZOS basic modules and the module defined by users. All ZOS basic modules are listed as the following:

system module

UTAL module

SIP module

RTP module

H323 module

test module

SUA module

### 2.3.2 Instance ID

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

### 2.3.3 Task ID

The task ID is used to identify different tasks. A task ID is a 32-bit unsigned integer consisting of a module ID and an instance ID, in which the higher 16bits are used for the module ID and the lower 16bits are used for instance ID. The reserved tasks are listed as the following:

Timer task

Log task

UTAL task

SIP task

RTP task

H323 task

Test task

SUA task

### 2.3.4 Protocol ID

Protocol ID is used to identify different protocols in ZOS ABNF module. These IDs are listed below by ZOS.

```
/* ZOS protocol type */
#define ZPROTOCOL_UNKNOWN 0 /* unknown protocol */
#define ZPROTOCOL_SDP 1 /* SDP protocol */
#define ZPROTOCOL_MGCP 2 /* MGCP text protocol */
#define ZPROTOCOL_MGCO_TXT 3 /* MEGACO text protocol */
#define ZPROTOCOL_MGCO_BIN 4 /* MEGACO binary protocol */
#define ZPROTOCOL_SIP 5 /* SIP protocol */
#define ZPROTOCOL_H323 6 /* H.323 protocol */
```

For SIP stack, the protocol ID should be **ZPROTOCOL\_SIP**.

## 2.4 SIP Primitives

The SUA and the SIP stack interact with each other via a group of functions known as primitives. These primitives are a set of well-defined functions, which take the form of:

Request Primitives

These primitives, provided by the SIP stack, are invoked by SUA to send requests to the SIP stack. The SIP stack will then send these requests to the receiver which is also a SIP stack. The request primitives are listed as the following:

[Sip\\_SendSimReq](#)

[Sip\\_SendSsmReq](#)

[Sip\\_SendSamReq](#)

[Sip\\_SendScmReq](#)

[Sip\\_SendSmmReq](#)

[Sip\\_SendStmReq](#)

[Sip\\_SendDamReq](#)

[Sip\\_SendCimReq](#)

#### Response Primitives

These primitives, provided by the SIP stack, are invoked by users to send responses to the SIP stack. The SIP stack will then send these responses to sender who sends the requests. These response primitives are listed as the following:

[Sip\\_SendSimRsp](#)

[Sip\\_SendSsmRsp](#)

[Sip\\_SendSmmRsp](#)

[Sip\\_SendDamRsp](#)

[Sip\\_SendCimRsp](#)

To learn more about the service primitives please refer to [Primitive Interfaces](#).

#### Indication Primitives

These primitives are invoked by the SIP stack into the SUA to indicate the reception of a request (such as an INVITE request) from another SIP stack of a remote system. The indication primitives are provided by the SUA.

#### Confirm Primitives

These primitives are invoked by the SIP stack into the SUA to confirm the reception of a response to an earlier request which has been completed. The confirm primitives are provided by the SUA.

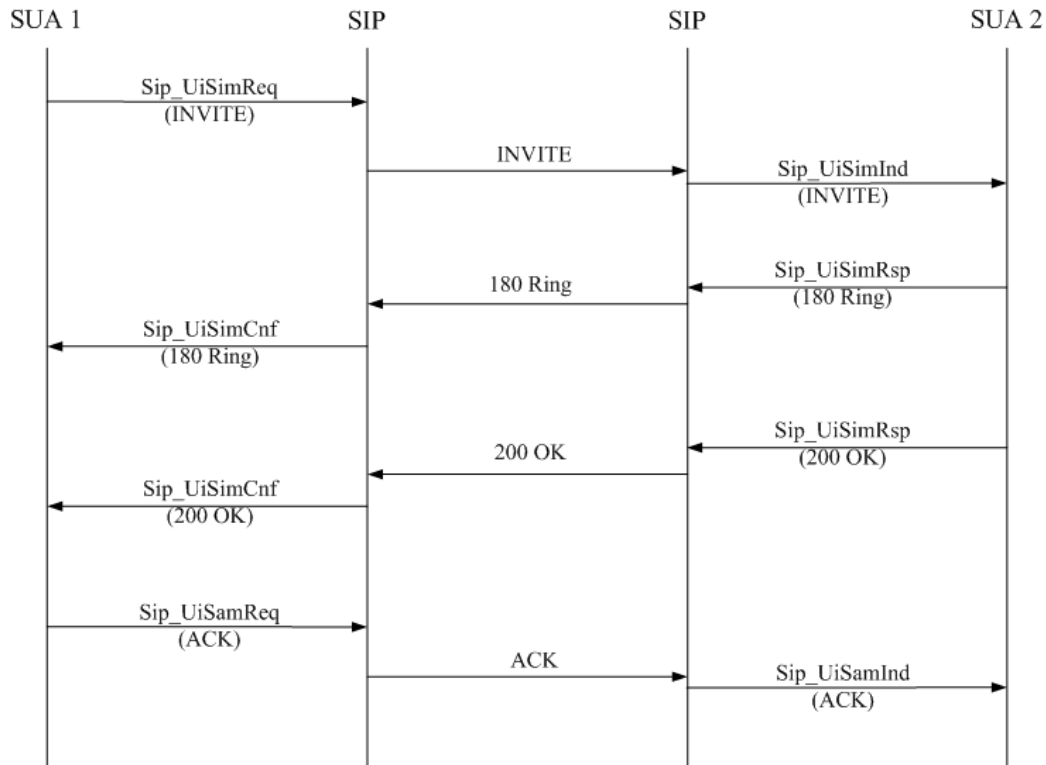


Figure 2-2 Use Primitives to Send Requests and Responses

Figure 2-4 depicts how to use primitives to send requests and responses. In the figure above, Sip\_SendsimReq() and Sip\_SendSamReq() are request primitives invoked by SUA1 to send requests to the SIP stack. Sip\_SendSimRsp() is a response primitive invoked by SUA2 to send responses. Sip\_SendSimCnf() is a confirm primitive invoked by the SIP stack to inform SUA1 of the reception of responses. Sip\_SendSimInd() is an indication primitive invoked by the SIP stack to inform SUA2 of the reception of requests.

## 3. User Interfaces

### 3.1 Interface Structures

#### 3.1.1 ZOS Structure

##### 3.1.1.1 *ST\_ZOS\_SSTR*

It is a structure containing a ZOS string.

```
typedef struct tagZOS_SSTR
{
    ZCHAR *pcStr;           /* string pointer */
    ZUSHORT wLen;          /* string length */
    ZUCHAR aucSpare[2];    /* for 32 bit alignment */
} ST_ZOS_SSTR;
```

##### 3.1.1.2 *ST\_ZOS\_DBUF*

The ZOS data buffer structure is used by ZOS memory mechanism.

```
typedef struct tagZOS_DBUF
{
    struct tagZOS_DBUF *pstNext; /* next data buffer */
    ZPOOLID zPoolId;            /* memory pool id for dbuf alloc */
    ZABUFID zAMemBuf;           /* aggregation memory buffer */
    ZXBUFID zXMemBuf;           /* exchange memory buffer */
    ZULONG dwBufLen;            /* buffer used length */
    ZULONG dwDftBlkSize;        /* default data block size in buffer */
    ZUCHAR ucBufType;           /* buffer mode ZDBUF_TYPE_BYTE... */
    ZUCHAR ucRefCnt;            /* buffer reference count */
    ZUCHAR aucSpare[2];         /* for 32 bit alignment */
#ifdef ZOS_SUPT_DUMP
    ZDUMPID zDumpId;            /* stack dump */
#endif
    ST_ZOS_DBUF_DATA *pstHead; /* the first data block in buffer */
    ST_ZOS_DBUF_DATA *pstTail; /* the last data block in buffer */
} ST_ZOS_DBUF;
```

##### 3.1.1.3 *ST\_ABNF\_MSG*

When a message is received and before being handled by SIP stack, it should be converted into a SIP message first. The message is decoded by ABNF. So before decoding, this message is an ABNF message.

```

typedef struct tagABNF_MSG
{
    ZULONG dwProtocol;           /* protocol type */
    ST_ZOS_DBUF *pstMemBuf;      /* data memory buffer */
    ST_ABNF_ERR_INFO *pstErr;    /* error info */
    ST_ABNF_BUF_INFO stBuf;      /* data buffer */
    ST_ABNF_BUF_INFO stSavedBuf; /* saved data buffer */
    ST_ABNF_CHR_INFO stChr;      /* character property */
    ST_ABNF_SEPA_INFO stSepa;    /* separator character info */
    ST_ABNF_EXTN_INFO stExtn;    /* extend info */
} ST_ABNF_MSG;

```

It is ABNF message structure defined in zos\_abnf\_type.h. Its members are described below:

Member	Description
dwProtocol	The protocol type.
pstMemBuf	A pointer to ZOS memory buffer.
pstErr	A pointer to the error information.
stBuf	A variable of ST_ABNF_BUF_INFO structure type, used to hold the ABNF data information.
stSavedBuf	A variable of ST_ABNF_BUF_INFO structure type, used to hold the information of saved data.
stChr	A variable of ST_ABNF_CHR_INFO structure type, used to hold the character properties.
stSepa	A variable of ST_ABNF_SEPA_INFO structure type, used to hold the separator information.
stExtn	A variable of ST_ABNF_EXTN_INFO structure type, used to hold the extended parameters. It supports 3 extended parameters at most.

Table 3-1 ST\_ABNF\_MSG

#### 3.1.1.4 ST\_ZOS\_MSP

ZOS post is used to deliver task messages between different system tasks.

```

typedef struct tagZOS_MSP
{
    ZCPUID zSendCpuId;           /* sender CPU ID */
    ZTASKID zSendTaskId;        /* sender task ID */
    ZCPUID zRecvCpuId;          /* receiver CPU ID */
    ZTASKID zRecvTaskId;        /* receiver task ID */
    ZEVENTID zEvtId;            /* event ID */
    ZPOOLID zPoolId;            /* memory pool ID */
} ST_ZOS_MSP;

```

Its members are described below:

Parameters	Description
zSendCpuId	Sender's CPU ID.
zSendTaskId	Sender's task ID.
zRevCpuId	Receiver's CPU ID.
zRecvTaskId	Receiver's task ID.
zEvtId	The event ID.
zPoolId	The memory pool ID.

Table 3-2 ST\_ZOS\_MSP

### 3.1.1.5 ST\_ZOS\_INET\_ADDR

```
typedef struct tagZOS_INET_ADDR
{
    ZUSHORT wType;           /* ZINET_IPV4... */
    ZUSHORT wPort;          /* not order[host or n/w] dependent */
    union
    {
        ZUINT iIp;          /* 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;
```

## 3.1.2 SIP Message Structures

### 3.1.2.1 ST\_SIP\_MSG

The SIP stack has its own message structure. Only the messages whose type is of ST\_SIP\_MSG( SIP message structure) can be handled by SIP stack.

```

typedef struct tagSIP_MSG
{
    ZUCHAR ucPres;           /* present flag */
    ZUCHAR ucReqPres;       /* Request present flag */
    ZUCHAR aucSpare[2];     /* for 32 bit alignment */
    ST_ZOS_DBUF *pstMemBuf; /* memory buffer */
    ST_ZOS_DBUF *pstMsgBuf; /* message buffer */
    ST_ZOS_SSTR stMsgStr;   /* message string(decode used) */
    union
    {
        ST_SIP_REQ_LINE stReqLine; /* Request Line */
        ST_SIP_STATUS_LINE stStatusLine; /* Status Line */
    } u;
    ST_SIP_MSG_HDR_LST stHdrLst; /* message-header list */
    ST_SIP_BODY stBody;         /* message-body */
} ST_SIP_MSG;

```

It is the structure of a SIP message which is decoded or to be encoded. Refer to RFC 3261 for detailed SIP message structure description. Its members are described below:

Member	Description
ucPres	Indicates whether the message is present.
ucReqPres	Indicates whether the message is a request or a response. If the message is a request then it has to be set to ZTRUE.
aucSpare	It is for 32-bit alignment.
pstMemBuf	A pointer to ZOS memory buffer.
pstMsgBuf	A pointer to the message buffer.
stMsgStr	A variable of tagZOS_SSTR structure type, used while decoding.
u.stReqLine	A variable of tagSIP_REQ_LINE structure type.
u.stStatusLine	A variable of tagSIP_STATUS_LINE structure type.
stHdrLst	A message header list.
stBody	A variable of tagSIP_BODY structure type.

Table 3-3 ST\_SIP\_MSG

### 3.1.2.2 ST\_SIP\_SESS\_EVNT

It is SIP session event structure.

```

typedef struct tagSIP_SESS_EVNT
{
    ZUCHAR ucEvtType;           /* event type EN_SIP_SESS_EVNT_TYPE */
    ZUCHAR ucEvtOwner;         /* event owner EN_SIP_EVNT_OWNER_TYPE */
    ZUCHAR ucMsgType;          /* message type EN_SIP_EVNT_MSG_TYPE */
    ZUCHAR ucMethodType;       /* method type EN_SIP_METHOD */
    ZULONG dwStatusCode;       /* status code */
    ZULONG dwCompId;           /* component id */
    ZULONG dwSessUserId;       /* session user id */
    ZULONG dwDlgUserId;        /* dialog user id */
    ZULONG dwTransUserId;      /* transaction user id */
    ZULONG dwSessId;           /* session id */
    ZULONG dwDlgId;            /* dialog id */
    ZULONG dwTransId;          /* transaction id */
    ZULONG dwReferExpires;     /* refer expires */
    ST_SIP_MSG *pstMsg;         /* sip message */
    ST_SIP_METHOD *pstMethod;   /* sip method */
    ST_SIP_EVNT_PKG *pstEvtPkg; /* event package for subscription */
    ST_SIP_TPT_ADDR stTptAddr;  /* transport address */
} ST_SIP_SESS_EVNT;

```

It is the structure of a SIP session event, including the event type, the event owner, the message type and whether the message is an outgoing message, the status code, the transport context, the user ID, the session ID, the dialog ID, the transaction ID, the SIP message.

Parameters	Description
ucEvtType	Indicates the event type.
ucEvtOwner	Indicates the event owner type.
ucMsgType	Indicates the message type.
ucMethodType	Indicates the method type.
dwStatusCode	The status code.
dwCompId	Component ID
dwSessUserId	The session user ID.
dwDlgUserId	The dialog user ID.
dwTransUserId	The transaction user ID.
dwSessId	The session ID.
dwDlgId	The dialog ID.
dwTransId	The transaction ID.
dwReferExpires	Expires value of REFER
pstMsg	A pointer to SIP message structure.
pstMethod	A pointer to SIP method structure.

---

pstEvtPkg	A pointer to event for subscription.
stTptAddr	The transport address.

Table 3-4 ST\_SIP\_SESS\_EVNT

### ***3.1.2.3 EN\_SIP\_SESS\_EVNT\_TYPE***

It is SIP session event type enum including all events that may happen during a session.

```
typedef enum EN_SIP_SESS_EVNT_TYPE
{
    /* session error message event */
    EN_SIP_SESSE_ERR_IND = 0,

    /* session initiating message event */
    EN_SIP_SESSE_SIM_CNF = 1,
    EN_SIP_SESSE_SIM_IND = 2,

    /* session status message event */
    EN_SIP_SESSE_SSM_CNF = 3,
    EN_SIP_SESSE_SSM_IND = 4,

    /* session acknowledge message event */
    EN_SIP_SESSE_SAM_IND = 5,

    /* session cancel message event */
    EN_SIP_SESSE_SCM_IND = 6,

    /* session modifying message event */
    EN_SIP_SESSE_SMM_CNF = 7,
    EN_SIP_SESSE_SMM_IND = 8,

    /* session terminating message event */
    EN_SIP_SESSE_STM_CNF = 9,
    EN_SIP_SESSE_STM_IND = 10,

    /* session dialog associate message event */
    EN_SIP_SESSE_DAM_CNF = 11,
    EN_SIP_SESSE_DAM_IND = 12,

    /* session call independent message event */
    EN_SIP_SESSE_CIM_CNF = 13,
    EN_SIP_SESSE_CIM_IND = 14,

    /* below message definition can be used by upper modules for
       exchange session event then call sip uri interfaces */
    /* session initiating message event */
    EN_SIP_SESSE_SIM_REQ = 20,
    EN_SIP_SESSE_SIM_RSP = 21,

    /* session status message event */
    EN_SIP_SESSE_SSM_REQ = 22,
    EN_SIP_SESSE_SSM_RSP = 23,
```

```

/* session acknowledge message event */
EN_SIP_SESSE_SAM_REQ = 24,

/* session cancel message event */
EN_SIP_SESSE_SCM_REQ = 25,

/* session modifying message event */
EN_SIP_SESSE_SMM_REQ = 26,
EN_SIP_SESSE_SMM_RSP = 27,

/* session terminating message event */
EN_SIP_SESSE_STM_REQ = 28,

/* session dialog associate message event */
EN_SIP_SESSE_DAM_REQ = 29,
EN_SIP_SESSE_DAM_RSP = 30,

/* session dialog call independent message event */
EN_SIP_SESSE_CIM_REQ = 31,
EN_SIP_SESSE_CIM_RSP = 32,

EN_SIP_SESSE_UNKNOWN = 33
} EN_SIP_SESS_EVNT_TYPE;

```

Each constant of the enum represents a session event. All the constants above are described below:

Constant	Description
EN_SIP_SESSE_ERR_IND	Error indication. Its value is 0.
Following type is for up stream event.	
EN_SIP_SESSE_SIM_CNF	Confirmation of receiving a response to an INVITE request.
EN_SIP_SESSE_SIM_IND	Indication of receiving an INVITE request.
EN_SIP_SESSE_SSM_CNF	Confirmation of receiving a response to a session status request.
EN_SIP_SESSE_SSM_IND	Indication of receiving a session status request.
EN_SIP_SESSE_SAM_IND	Indication of receiving an ACK request.
EN_SIP_SESSE_SCM_IND	Indication of receiving a CANCEL request.
EN_SIP_SESSE_SMM_CNF	Confirmation of receiving a response to a re-INVITE request.
EN_SIP_SESSE_SMM_IND	Indication of receiving a re-INVITE request.
EN_SIP_SESSE_STM_CNF	Confirmation of receiving a response to a BYE request.
EN_SIP_SESSE_STM_IND	Indication of receiving a BYE request.
EN_SIP_SESSE_DAM_CNF	Confirmation of receiving a response to a dialog-associated message request.
EN_SIP_SESSE_DAM_IND	Indication of receiving a dialog-associated message request.

EN_SIP_SESSE_CIM_CNF	Confirmation of receiving a response to a call independent request.
EN_SIP_SESSE_CIM_IND	Indication of receiving a call independent request.
Following type for down stream event.	
EN_SIP_SESSE_SIM_REQ	Generation of an INVITE request.
EN_SIP_SESSE_SIM_RSP	Generation of a response to an INVITE request.
EN_SIP_SESSE_SSM_REQ	Generation of a session status request.
EN_SIP_SESSE_SSM_RSP	Generation of a response to a session status request.
EN_SIP_SESSE_SAM_REQ	Generation of an ACK request.
EN_SIP_SESSE_SCM_REQ	Generation of a CANCEL request.
EN_SIP_SESSE_SMM_REQ	Generation of a re-INVITE request.
EN_SIP_SESSE_SMM_RSP	Confirmation of receiving a response to a re-INVITE request.
EN_SIP_SESSE_STM_REQ	Generation of a BYE request.
EN_SIP_SESSE_DAM_REQ	Generation of a dialog-associated message request.
EN_SIP_SESSE_DAM_RSP	Generation of a response to a dialog-associated request.
EN_SIP_SESSE_CIM_REQ	Generation of a call independent request.
EN_SIP_SESSE_CIM_RSP	Generation of a response to a call independent request.
EN_SIP_SESSE_UNKNOWN	An unknown session event.

Table 3-5 EN\_SIP\_SESS\_EVNT\_TYPE

### 3.1.2.4 ST\_SIP\_HDR\_CONTENT\_TYPE

```
typedef struct tagSIP_HDR_CONTENT_TYPE
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR aucSpare[3];          /* for 32 bit alignment */
    ST_SIP_MEDIA_TYPE stMediaType; /* media-type */
} ST_SIP_HDR_CONTENT_TYPE;
```

### 3.1.2.5 ST\_SIP\_HDR\_VIA

```
typedef struct tagSIP_HDR_VIA
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR aucSpare[3];          /* for 32 bit alignment */
    ST_SIP_VIA_PARM_LST stParmLst; /* via-param list */
} ST_SIP_HDR_VIA;
```

**3.1.2.6 ST\_SIP\_SIP\_URI**

```

typedef struct tagSIP_SIP_URI
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR ucUserInfoPres;        /* userinfo present flag */
    ZUCHAR ucHdrsPres;            /* Headers present flag */
    ZUCHAR aucSpare[1];           /* for 32 bit alignment */
    ST_SIP_USER_INFO stUserInfo; /* userinfo */
    ST_SIP_HOST_PORT stHostPort; /* hostport */
    ST_SIP_URI_PARM_LST stUriParmLst; /* uri-parameters */
    ST_SIP_HDRS stHdrs;           /* Headers */
} ST_SIP_SIP_URI;

```

**3.1.2.7 ST\_SIP\_TEL\_URI**

```

typedef struct tagSIP_TEL_URI
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR ucGlobalNumberPres;    /* global-number present flag */
    ZUCHAR aucSpare[2];           /* for 32 bit alignment */
    union
    {
        ST_SIP_GLOBAL_NUMBER stGlobal; /* global-number */
        ST_SIP_LOCAL_NUMBER stLocal; /* local-number */
    } u;
} ST_SIP_TEL_URI;

```

**3.1.2.8 ST\_SIP\_IM\_URI**

```

typedef struct tagSIP_IM_URI
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR ucDescPres;            /* [ to ] [ headers ] present flag */
    ZUCHAR aucSpare[2];           /* for 32 bit alignment */
    ST_ZOS_SSTR stDesc;           /* [ to ] [ headers ] in rfc3860 */
} ST_SIP_IM_URI;

```

### 3.1.2.9 *ST\_SIP\_CALLID*

```
typedef struct tagSIP_CALLID
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR ucHostPres;           /* host word present flag */
    ZUCHAR aucSpare[2];          /* for 32 bit alignment */
    ST_ZOS_SSTR stNumber;        /* number word */
    ST_ZOS_SSTR stHost;          /* host word */
} ST_SIP_CALLID;
```

### 3.1.2.10 *ST\_SIP\_HDR\_CSEQ*

```
typedef struct tagSIP_HDR_CSEQ
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR aucSpare[3];          /* for 32 bit alignment */
    ZULONG dwCseqVal;            /* CSeq value */
    ST_SIP_METHOD stMethod;      /* Method */
} ST_SIP_HDR_CSEQ;
```

### 3.1.2.11 *ST\_SIP\_HDR\_RACK*

```
typedef struct tagSIP_HDR_RACK
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR aucSpare[3];          /* for 32 bit alignment */
    ZULONG dwRspNum;             /* response-num */
    ZULONG dwCseqNum;            /* CSeq-num */
    ST_SIP_METHOD stMethod;      /* Method */
} ST_SIP_HDR_RACK;
```

**3.1.2.12 ST\_SIP\_HDR\_FROM\_TO**

```

typedef struct tagSIP_HDR_FROM_TO
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR ucNameAddrPres;        /* name-addr present flag */
    ZUCHAR aucSpare[2];           /* for 32 bit alignment */
    ST_SIP_FROM_TO_PARM_LST stParmLst; /* from-param list */
    union
    {
        ST_SIP_NAME_ADDR stNameAddr; /* name-addr */
        ST_SIP_ADDR_SPEC stAddrSpec; /* addr-spec */
    } u;
} ST_SIP_HDR_FROM_TO;

```

**3.1.2.13 ST\_SIP\_HDR\_REFERER\_TO**

```

typedef struct tagSIP_HDR_REFERER_TO
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR ucNameAddrPres;        /* name-addr present flag */
    ZUCHAR aucSpare[2];           /* for 32 bit alignment */
    ST_SIP_GEN_PARM_LST stParmLst; /* generic-param list */
    union
    {
        ST_SIP_NAME_ADDR stNameAddr; /* name-addr */
        ST_SIP_ADDR_SPEC stAddrSpec; /* addr-spec */
    } u;
} ST_SIP_HDR_REFERER_TO;

```

**3.1.2.14 ST\_SIP\_HDR\_REFERRED\_BY**

```

typedef struct tagSIP_HDR_REFERRED_BY
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR ucNameAddrPres;        /* name-addr present flag */
    ZUCHAR aucSpare[2];           /* for 32 bit alignment */
    ST_SIP_GEN_PARM_LST stParmLst; /* generic-param list */
    union
    {
        ST_SIP_NAME_ADDR stNameAddr; /* name-addr */
        ST_SIP_ADDR_SPEC stAddrSpec; /* addr-spec */
    } u;
} ST_SIP_HDR_REFERRED_BY;

```

**3.1.2.15 ST\_SIP\_HDR\_USER\_AGENT**

```
typedef struct tagSIP_HDR_USER_AGENT
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR aucSpare[3];           /* for 32 bit alignment */
    ST_SIP_SERVER_VAL stVal;      /* server-val */
    ST_SIP_SERVER_VAL_LST stValLst; /* server-val list */
} ST_SIP_HDR_USER_AGENT;
```

**3.1.2.16 ST\_SIP\_HDR\_SERVER**

```
typedef struct tagSIP_HDR_SERVER
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR aucSpare[3];           /* for 32 bit alignment */
    ST_SIP_SERVER_VAL stVal;      /* server-val */
    ST_SIP_SERVER_VAL_LST stValLst; /* server-val list */
} ST_SIP_HDR_SERVER;
```

**3.1.2.17 ST\_SIP\_HDR\_EVNT**

```
typedef struct tagSIP_HDR_EVNT
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR aucSpare[3];           /* for 32 bit alignment */
    ST_SIP_EVNT_TYPE stEvntType; /* event-type */
    ST_SIP_EVNT_PARM_LST stParmLst; /* event-param list */
} ST_SIP_HDR_EVNT;
```

**3.1.2.18 ST\_SIP\_HDR\_REPLACES**

```
typedef struct tagSIP_HDR_REPLACES
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR aucSpare[3];           /* for 32 bit alignment */
    ST_SIP_CALLID stCallid;       /* callid */
    ST_SIP_REPLACES_PARM_LST stParmLst; /* replaces-param list */
} ST_SIP_HDR_REPLACES;
```

### 3.1.2.19 ST\_SIP\_EVNT\_PKG

```
typedef struct tagSIP_EVNT_PKG
{
    ZUCHAR ucPres;           /* present flag */
    ZUCHAR ucType;          /* package type EN_SIP_EVNT_PKG */
    ZUCHAR aucSpare[2];     /* for 32 bit alignment */
    ST_ZOS_SSTR stOther;    /* other package */
} ST_SIP_EVNT_PKG;
```

### 3.1.2.20 ST\_SIP\_HDR\_SUBS\_STA

```
typedef struct tagSIP_HDR_SUBS_STA
{
    ZUCHAR ucPres;           /* present flag */
    ZUCHAR aucSpare[3];     /* for 32 bit alignment */
    ST_SIP_SUBSTA_VAL stSubsVal; /* substate-value */
    ST_SIP_SUBEXP_PARAMS_LST stParmLst; /* subexp-params list */
} ST_SIP_HDR_SUBS_STA;
```

### 3.1.2.21 ST\_SIP\_HDR\_ALLOW

```
typedef struct tagSIP_HDR_ALLOW
{
    ZUCHAR ucPres;           /* present flag */
    ZUCHAR aucSpare[3];     /* for 32 bit alignment */
    ST_SIP_METHOD stMethod; /* Method */
    ST_SIP_METHOD_LST stMethodLst; /* Method list */
} ST_SIP_HDR_ALLOW;
```

### 3.1.2.22 ST\_SIP\_METHOD

```
typedef struct tagSIP_METHOD
{
    ZUCHAR ucPres;           /* present flag */
    ZUCHAR ucType;          /* method type EN_SIP_METHOD */
    ZUCHAR aucSpare[2];     /* for 32 bit alignment */
    ST_ZOS_SSTR stExt;      /* extension or unknown type string */
} ST_SIP_METHOD;
```

**3.1.2.23 ST\_SIP\_HDR\_SUPTED**

```
typedef struct tagSIP_HDR_SUPTED
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR aucSpare[3];           /* for 32 bit alignment */
    ST_SIP_OPT_TAG_LST stOptTagLst; /* option-tag list */
} ST_SIP_HDR_SUPTED;
```

**3.1.2.24 ST\_SIP\_HDR\_REQUIRE**

```
typedef struct tagSIP_HDR_REQUIRE
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR aucSpare[3];           /* for 32 bit alignment */
    ST_SIP_OPT_TAG_LST stOptTagLst; /* option-tag list */
} ST_SIP_HDR_REQUIRE;
```

**3.1.2.25 ST\_SIP\_HOST\_PORT**

```
typedef struct tagSIP_HOST_PORT
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR ucPortPres;           /* Port present flag */
    ZUCHAR aucSpare[2];           /* for 32 bit alignment */
    ST_SIP_HOST stHost;          /* host */
    ZULONG dwPort;               /* port */
} ST_SIP_HOST_PORT;
```

**3.1.2.26 ST\_SIP\_REQ\_URI**

```

typedef struct tagSIP_REQ_URI
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR ucType;                /* Request-URI type EN_SIP_REQ_URI */
    ZUCHAR aucSpare[2];          /* for 32 bit alignment */
    union
    {
        ST_SIP_SIP_URI stSipUri; /* SIP-URI */
        ST_SIP_SIP_URI stSipsUri; /* SIPS-URI */
        ST_SIP_IM_URI stImUri;    /* IM-URI rfc3860 */
        ST_SIP_ABSO_URI stAbsoUri; /* absoluteURI */
        ST_SIP_TEL_URI stTelUri; /* TEL-URI */
    } u;
} ST_SIP_REQ_URI;

```

**3.1.2.27 ST\_SIP\_ADDR\_SPEC**

```

typedef struct tagSIP_ADDR_SPEC
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR ucType;                /* addr-spec type EN_SIP_ADDR_SPEC */
    ZUCHAR aucSpare[2];          /* for 32 bit alignment */
    union
    {
        ST_SIP_SIP_URI stSipUri; /* SIP-URI */
        ST_SIP_SIP_URI stSipsUri; /* SIPS-URI */
        ST_SIP_TEL_URI stTelUri; /* TEL-URI */
        ST_SIP_IM_URI stImUri;    /* IM-URI rfc3860 */
        ST_SIP_IM_URI stMailtoUri; /* MAILTO-URI rfc2368 */
        ST_SIP_MC_URI stMidUri;   /* Message-ID uri */
        ST_SIP_MC_URI stCidUri;   /* Content-ID uri */
        ST_SIP_ABSO_URI stAbsoUri; /* absoluteURI */
    } u;
} ST_SIP_ADDR_SPEC;

```

**3.1.2.28 ST\_SIP\_CONTACT\_PARM**

```

typedef struct tagSIP_CONTACT_PARM
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR ucNameAddrPres;        /* name-addr present flag */
    ZUCHAR aucSpare[2];           /* for 32 bit alignment */
    ST_SIP_CONTACT_PARAMS_LST stParamsLst; /* contact-params list */
    union
    {
        ST_SIP_NAME_ADDR stNameAddr; /* name-addr */
        ST_SIP_ADDR_SPEC stAddrSpec; /* addr-spec */
    } u;
} ST_SIP_CONTACT_PARM;

```

**3.1.2.29 ST\_SIP\_HOST\_PORT**

```

typedef struct tagSIP_HOST_PORT
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR ucPortPres;            /* Port present flag */
    ZUCHAR aucSpare[2];           /* for 32 bit alignment */
    ST_SIP_HOST stHost;           /* host */
    ZULONG dwPort;                /* port */
} ST_SIP_HOST_PORT;

```

**3.1.2.30 ST\_SIP\_VIA\_PARM**

```

typedef struct tagSIP_VIA_PARM
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR aucSpare[3];           /* for 32 bit alignment */
    ST_SIP_SENT_PROTOCOL stProtocol; /* sent-protocol */
    ST_SIP_SENT_BY stBy;          /* sent-by */
    ST_SIP_VIA_PARAMS_LST stParamsLst; /* via-params list */
} ST_SIP_VIA_PARM;

```

**3.1.2.31 ST\_SIP\_EVNT\_TYPE**

```

typedef struct tagSIP_EVNT_TYPE
{
    ST_SIP_EVNT_PKG stEvtntPkg;    /* event package */
    ST_SIP_EVNT_TEMP_LST stEvtntTempLst; /* event-template */
} ST_SIP_EVNT_TYPE;

```

**3.1.2.32 ST\_SIP\_DISP\_NAME**

```

typedef struct tagSIP_DISP_NAME
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR ucNamePres;           /* name present flag */
    ZUCHAR aucSpare[2];          /* for 32 bit alignment */
    union
    {
        ST_ZOS_SSTR stName;      /* token name */
        ST_ZOS_SSTR stQStr;      /* quoted-string */
    } u;
} ST_SIP_DISP_NAME;

```

**3.1.2.33 ST\_SIP\_CREDENTIALS**

```

typedef struct tagSIP_CREDENTIALS
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR ucDigestRspPres;       /* digest-response present flag */
    ZUCHAR aucSpare[2];          /* for 32 bit alignment */
    union
    {
        ST_SIP_DIGEST_RSP stDigestRsp; /* digest-response */
        ST_SIP_OTHER_RSP stOther;      /* other-response */
    } u;
} ST_SIP_CREDENTIALS;

```

**3.1.2.34 ST\_SIP\_CHALLENGE**

```

typedef struct tagSIP_CHALLENGE
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR ucDigestPres;          /* digest present flag */
    ZUCHAR aucSpare[2];          /* for 32 bit alignment */
    union
    {
        ST_SIP_DIGEST stDigest;      /* digest */
        ST_SIP_OTHER_CHALLENGE stOther; /* other-challenge */
    } u;
} ST_SIP_CHALLENGE;

```

**3.1.2.35 ST\_SIP\_DIGEST\_RSP**

```
typedef struct tagSIP_DIGEST_RSP
{
    ST_SIP_DIG_RSP_LST stDigRspLst; /* dig-resp list */
} ST_SIP_DIGEST_RSP;
```

**3.1.2.36 ST\_SIP\_ALGO**

```
typedef struct tagSIP_ALGO
{
    ZUCHAR ucPres; /* present flag */
    ZUCHAR ucAkaNsPres; /* aka-namespace present flag */
    ZUCHAR aucSpare[2]; /* for 32 bit alignment */
    union
    {
        ST_SIP_AKA_NS stAkaNs; /* aka-namespace */
        ST_SIP_ALGO_VAL stAlgoVal; /* algorithm-value */
    } u;
} ST_SIP_ALGO;
```

**3.1.2.37 ST\_SIP\_HDR\_EXT\_HDR**

```
typedef struct tagSIP_HDR_EXT_HDR
{
    ZUCHAR ucPres; /* present flag */
    ZUCHAR aucSpare[3]; /* for 32 bit alignment */
    ST_ZOS_SSTR stName; /* header-name */
    ST_ZOS_SSTR stVal; /* header-value */
} ST_SIP_HDR_EXT_HDR;
```

**3.1.2.38 ST\_SIP\_CONTACT\_PARM**

```

typedef struct tagSIP_CONTACT_PARM
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR ucNameAddrPres;        /* name-addr present flag */
    ZUCHAR aucSpare[2];           /* for 32 bit alignment */
    ST_SIP_CONTACT_PARAMS_LST stParamsLst; /* contact-params list */
    union
    {
        ST_SIP_NAME_ADDR stNameAddr; /* name-addr */
        ST_SIP_ADDR_SPEC stAddrSpec; /* addr-spec */
    } u;
} ST_SIP_CONTACT_PARM;

```

**3.1.2.39 ST\_SIP\_AC\_VAL**

```

typedef struct tagSIP_AC_VAL
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR aucSpare[3];           /* for 32 bit alignment */
    ST_SIP_AC_PARAMS_LST stParamsLst; /* ac-params list */
} ST_SIP_AC_VAL;

```

**3.1.2.40 ST\_SIP\_MEDIA\_TYPE**

```

typedef struct tagSIP_MEDIA_TYPE
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR ucMtype;               /* m-type type EN_SIP_MTYPE */
    ZUCHAR ucMSubtype;           /* m-subtype type EN_SIP_MSUBTYPE */
    ZUCHAR aucSpare[1];           /* for 32 bit alignment */
    ST_ZOS_SSTR stMtypeExt;       /* m-type extension-token */
    ST_ZOS_SSTR stMSubTypeExt;    /* m-subtype extension-token */
    ST_SIP_MPARAM_LST stParmLst;  /* m-parameter list */
} ST_SIP_MEDIA_TYPE;

```

**3.1.2.41 ST\_SIP\_OPT\_TAG**

```
typedef struct tagSIP_OPT_TAG
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR ucType;                /* option-tag type EN_SIP_OPT_TAG */
    ZUCHAR aucSpare[2];          /* for 32 bit alignment */
    ST_ZOS_SSTR stOther;         /* other option-tag */
} ST_SIP_OPT_TAG;
```

**3.1.2.42 ST\_SIP\_BODY\_MPART**

```
typedef struct tagSIP_BODY_MPART
{
    ST_SIP_MSG_HDR_LST stHdrLst; /* message-header list */
    struct tagSIP_BODY *pstBody; /* body in multipart body */
} ST_SIP_BODY_MPART;
```

**3.1.2.43 ST\_SIP\_HDR\_PRIVACY**

```
typedef struct tagSIP_HDR_PRIVACY
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR aucSpare[3];          /* for 32 bit alignment */
    ST_SIP_PRIV_VAL stVal;        /* priv-value */
    ST_SIP_PRIV_VAL_LST stValLst; /* priv-value list */
} ST_SIP_HDR_PRIVACY;
```

**3.1.2.44 ST\_SIP\_STATUS\_LINE**

```
typedef struct tagSIP_STATUS_LINE
{
    ZUCHAR ucPres;                /* present flag */
    ZUCHAR aucSpare[3];          /* for 32 bit alignment */
    ST_SIP_VER stVer;            /* SIP-Version */
    ZULONG dwCode;               /* Status-Code */
    ST_ZOS_SSTR stReason;        /* Reason-Phrase */
} ST_SIP_STATUS_LINE;
```

**3.1.2.45 PFN\_SIPNTFYSESSEVNT**

```
typedef ZINT (*PFN_SIPNTFYSESSEVNT) (ST\_SIP\_SESS\_EVNT *pstEvnt);
```

### 3.1.3 SIP Headers and Associated Sub-Structures

The table below lists the headers and their associated sub-structures.

Header	Associated Structures
Defined in RFC 3261	
Accept	ST_SIP_HDR_ACPT
Accept-Encoding	ST_SIP_HDR_ACPT_ENCODING
Accept-Language	ST_SIP_HDR_ACPT_LANG
Alert-Info	ST_SIP_HDR_ALERT_INFO
Allow	ST_SIP_HDR_ALLOW
Authentication-Info	ST_SIP_HDR_AUTHEN_INFO
Authorization	ST_SIP_HDR_AUTHOR
Call-ID	ST_SIP_HDR_CALL_ID
Call-Info	ST_SIP_HDR_CALL_INFO
Contact	ST_SIP_HDR_CONTACT
Content-Disposition	ST_SIP_HDR_CONTENT_DISP
Content-Encoding	ST_SIP_HDR_CONTENT_ENCODING
Content-Language	ST_SIP_HDR_CONTENT_LANG
Content-Length	ZULONG
Content-Type	ST_SIP_HDR_CONTENT_TYPE
CSeq	ST_SIP_HDR_CSEQ
Date	ST_SIP_HDR_DATE
Error-Info	ST_SIP_HDR_ERROR_INFO
Expires	ZULONG
In-Reply-To	ST_SIP_HDR_IN_REPLY_TO
From or To	ST_SIP_HDR_FROM_TO
Max-Forwards	ZULONG
MIME-Version	ST_SIP_HDR_MIME_VER
Min-Expires	ZULONG
Organization	ST_SIP_HDR_ORGANIZATION
Priority	ST_SIP_HDR_PRIORITY
Proxy-Authenticate	ST_SIP_HDR_PROXY_AUTHEN
Proxy-Authorization	ST_SIP_HDR_PROXY_AUTHOR
Proxy-Require	ST_SIP_HDR_PROXY_REQUIRE
Record-Route	ST_SIP_HDR_ROUTE
Reply-To	ST_SIP_HDR_REPLY_TO
Require	ST_SIP_HDR_REQUIRE
Retry-After	ST_SIP_HDR_RETRY_AFTER

Route	ST_SIP_HDR_ROUTE
Server	ST_SIP_HDR_SERVER
Subject	ST_SIP_HDR_SUBJECT
Supported	ST_SIP_HDR_SUPTED
Timestamp	ST_SIP_HDR_TIMESTAMP
Unsupported	ST_SIP_HDR_UNSUPTED
User-Agent	ST_SIP_HDR_USER_AGENT
Via	ST_SIP_HDR_VIA
Warning	ST_SIP_HDR_WARNING
WWW-Authenticate	ST_SIP_HDR_WWW_AUTHEN
Defined in RFC 3262	
RAck	ST_SIP_HDR_RACK
RSeq	ST_SIP_HDR_RSEQ
Defined in RFC 3265	
Event	ST_SIP_HDR_EVNT
Allow-Events	ST_SIP_HDR_ALLOW_EVNTS
Subscription-State	ST_SIP_HDR_SUBS_STA
Defined in RFC 3515	
Refer-To	ST_SIP_HDR_REFERER_TO
Defined in RFC 3892	
Referred-By	ST_SIP_HDR_REFERRED_BY
Defined in RFC 3891	
Replaces	ST_SIP_HDR_REPLACES
Defined in RFC4028	
Session-Expires	ST_SIP_HDR_SESS_EXPIRES
Min-SE	ST_SIP_HDR_MIN_SE
Defined in RFC 3841	
Request-Disposition	ST_SIP_HDR_REQ_DISP
Accept-Contact	ST_SIP_HDR_ACPT_CONTACT
Reject-Contact	ST_SIP_HDR_REJ_CONTACT
Defined in draft-ietf-sip-privacy	
Anonymity	ST_SIP_HDR_ANONY
RPID-Privacy	ST_SIP_HDR_RPID_PRIVACY
Remote-Party-ID	ST_SIP_HDR_REMOTE_PARTY_ID
Defined in RFC 3903	
SIP-ETag	ST_SIP_HDR_SIP_ETAG
SIP-If-Match	ST_SIP_HDR_SIP_IF_MATCH
Defined in RFC 3911	

Join	ST_SIP_HDR_JOIN (rfc3911)
Defined in RFC 3455	
P-Associated-URI	ST_SIP_HDR_P_ASO_URI
P-Called-Party-ID	ST_SIP_HDR_P_CALLED_PTY_ID
P-Visited-Network-ID	ST_SIP_HDR_P_VNET_ID
P-Access-Network-Info	ST_SIP_HDR_P_ACCESS_NET_INFO
P-Charging-Addr	ST_SIP_HDR_P_CHARG_ADDR
P-Charging-Vector	ST_SIP_HDR_P_CHARG_VEC
Defined in RFC 3327	
Path	ST_SIP_HDR_ROUTE
Defined in RFC 3608	
Service-Route	ST_SIP_HDR_ROUTE
Defined in RFC 3325	
PAssertedID	ST_SIP_HDR_P_ASSERTED_ID
PPreferredID	ST_SIP_HDR_P_PREFERRED_ID
Defined in RFC 3313	
P-Media-Authorization	ST_SIP_HDR_P_MEDIA_AUTHOR
Defined in RFC 3323	
Privacy-hdr	ST_SIP_HDR_PRIVACY
Defined in RFC 3326	
Reason	ST_SIP_HDR_REASON
Defined in RFC 3329	
security-client	ST_SIP_HDR_SEC_CLI
security-server	ST_SIP_HDR_SEC_SERV
security-verify	ST_SIP_HDR_SEC_VERIFY
Defined in RFC 3603	
P-DCS-Trace-Party-ID	ST_SIP_HDR_P_DCS_TRACE_PTY_ID
P-DCS-OSPS	ST_SIP_HDR_P_DCS_OSPS
P-DCS-Billing-Info	ST_SIP_HDR_P_DCS_BILL_INFO
P-DCS-LAES	ST_SIP_HDR_P_DCS_LAES
P-DCS-Redirect	ST_SIP_HDR_P_DCS_REDIR
Defined in RFC 4244	
History-Info	ST_SIP_HDR_HI_INFO
Defined in draft poc-p-headers	
P-Alerting-Mode	ST_SIP_HDR_P_ALERT_MODE
P-Answer-State	ST_SIP_HDR_P_ANSWER_STATE
Defined in RFC 4457	
P-User-Database	ST_SIP_HDR_P_USER_DB

Defined in RFC 4488	
Refer-Sub	ST_SIP_HDR_REFERER_SUB
Defined in RFC 2392	
Message-ID	ST_SIP_HDR_MSG_ID
Content-ID	ST_SIP_HDR_CONTENT_ID
Defined in draft-ietf-sip-answer-mode	
Answer-Mode	ST_SIP_HDR_ANSWER_MODE
Priv-Answer-Mode	ST_SIP_HDR_PRIV_ANSWER_MODE
Defined in draft-drage-sipping-service-identification	
PAssertedService	ST_SIP_HDR_P_ASSERTED_SRV
PPreferredService	ST_SIP_HDR_P_PREFERRED_SRV
For extension	
Extension-Header	ST_SIP_HDR_EXT_HDR

Table 3-6 Headers and Associated Sub-Structure

### 3.1.4 SIP Configuration Interfaces

The access interface with name under following rules:

Set interface is ZINT Sip\_CfgSet+Name(Type).

Get interface is ZINT Sip\_CfgGet+Name(Type \*).

The access interface for TaskPriority is

ZINT Sip\_CfgSetTaskPriority(ZINT); and

ZINT Sip\_CfgGetTaskPriority(ZINT \*).

Name	Type	Description
TaskPriority	ZINT	SIP task priority.
TaskStackSize	ZULONG	SIP task stack size in byte.
TaskQueueSize	ZULONG	SIP task queue size.
TaskTimerNum	ZULONG	SIP task timer number.
UpperTaskId	ZTASKID	The upper task ID to which send up event.
IsProxy	ZBOOL	Whether SIP stack work as a proxy.
NtfySessEvt	PFN_SIPNTFYSESSEVNT	The callback function to send session event.
NtfyProxyEvt	PFN_SIPNTFYPROXYEVNT	The callback function to send proxy event.
MaxForwards	ZULONG	Max forward number in SIP message.
MtuSize	ZULONG	MTU for UDP transport.
ProductVal	ZCHAR *	Product value string.
LogLevel	ZULONG	Log level in SIP stack.
IsIpv6Tpt	ZBOOL	Whether SIP stack work on IPv6 mode.
UServAddr	ST_ZOS_INET_ADDR *	The UDP binding address of SIP stack.
TServAddr	ST_ZOS_INET_ADDR *	The TCP binding address of SIP stack.

TcpConnTimes	ZULONG	The retry times count when TCP connect failed.
TlsServAddr	ST_ZOS_INET_ADDR *	The TLS binding address of SIP stack.
IpTosVal	ZUCHAR	The TOS value using TLS.
T1	ZULONG	The time length of Timer 1.
T2	ZULONG	The time length of Timer 2.
T4	ZULONG	The time length of Timer 4.
TA	ZULONG	The time length of Timer A.
TB	ZULONG	The time length of Timer B.
TC	ZULONG	The time length of Timer C.
TD	ZULONG	The time length of Timer D.
TE	ZULONG	The time length of Timer E.
TF	ZULONG	The time length of Timer F.
TG	ZULONG	The time length of Timer G.
TH	ZULONG	The time length of Timer H.
TI	ZULONG	The time length of Timer I.
TJ	ZULONG	The time length of Timer J.
TK	ZULONG	The time length of Timer K.
TL	ZULONG	The time length of Timer L.
WaitNtfyTime	ZULONG	The time length of wait NOTIFY after SUBSCRIBE succeed.
WaitConnTime	ZULONG	The time length of wait establish TCP connection.
CallNum	ZULONG	The max number of call simultaneously.
SessNum	ZULONG	The max number of session simultaneously.
DlgNum	ZULONG	The max number of dialog simultaneously.
SubsdNum	ZULONG	The max number of subscription simultaneously.
TransNum	ZULONG	The max number of transaction simultaneously.
ConnNum	ZULONG	The max number of connection simultaneously.
TlsMethod	ZULONG	The type of TLS.
TlsCertFile	ZCHAR *	The certification file of TLS.
TlsTrustCertFile	ZCHAR *	The trust certification file of TLS.
TlsPrvKeyFile	ZCHAR *	The private key file of TLS.
TlsPrvKeyPass	ZCHAR *	The private key password of TLS.

Table 3-7 Properties Description

## 3.2 ABNF Interfaces

### 3.2.1 Sip\_AbnfInit

Initiates the SIP ABNF resource. The SIP ABNF resource must be ready before using SIP ABNF to decode and encode functions.

```
ZINT Sip_AbnfInit();
```

[Parameters]

**Input parameters:**

None.

**Output parameters:**

None.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

### 3.2.2 Sip\_AbnfDestroy

It is used to destroy the SIP ABNF resource.

```
ZVOID Sip_AbnfDestroy();
```

**[Parameters]****Input parameters:**

None.

**Output parameters:**

None.

**[Return value]**

None.

### 3.2.3 Sip\_DecodeMsg

It is responsible for decoding SIP messages.

```
ZINT Sip_DecodeMsg(ST_ABNF_MSG *pstAbnfMsg, ST_SIP_MSG *pstSipMsg);
```

**[Parameters]****Input parameters:**

[ST\\_ABNF\\_MSG](#) \*pstAbnfMsg

A pointer to ABNF message structure. This message is going to be decoded by ABNF. This ABNF message should be initialized by `Abnf_MsgInit()`. Users should provide memory buffer, error info, packet data string, and protocol type. See example for detailed instructions.

**Output parameters:**

[ST\\_SIP\\_MSG](#) \*pstSipMsg

A pointer to a SIP message which is the result of a decoded ABNF message.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

For instructions on how to use this interface please see the [example](#).

### 3.2.4 Sip\_EncodeMsg

Encodes a SIP message to an ABNF message.

```
ZINT Sip_EncodeMsg(ST ABNF MSG *pstAbnfMsg, ST SIP MSG *pstSipMsg);
```

[Parameters]

#### Input parameters:

[ST ABNF MSG](#) \*pstAbnfMsg

A pointer to an ABNF message.

This ABNF message should be initialized by `Abnf_MsgInit()`. Users should provide memory buffer, error info, and protocol type. See examples for detailed instructions.

[ST SIP MSG](#) \*pstSipMsg

A pointer to a SIP message which is to be encoded by ABNF.

#### Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

For instructions on how to use this interface please see the [example](#).

### 3.2.5 Sip\_CreateMsgHdr

Creates a message header of a specific type and returns the address which is the union member of `ST_SIP_MSG_HDR`.

```
ZVOID * Sip_CreateMsgHdr(ST SIP MSG *pstMsg, ZUCHAR ucType);
```

[Parameters]

#### Input parameters:

[ST SIP MSG](#) \*pstMsg

The message.

ZUCHAR ucType

Type of the header.

#### Output parameters:

None.

[Return value]

Returns the address of the header on success, or ZNULL.

### 3.2.6 Sip\_DeleteMsgHdr

Deletes the first message header of a specific type.

```
ZINT Sip_DeleteMsgHdr(ST\_SIP\_MSG *pstMsg, ZUCHAR ucType);
```

[Parameters]

**Input parameters:**

[ST\\_SIP\\_MSG](#) \*pstMsg

The message.

ZUCHAR ucType

Type of the header.

**Output parameters:**

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.2.7 Sip\_FillMsgHdr

Fills a message header of a particular type with a string value.

```
ZINT Sip_FillMsgHdr(ST\_SIP\_MSG *pstMsg, ZUCHAR ucType,  
                   ST\_ZOS\_SSTR *pstStr);
```

[Parameters]

**Input parameters:**

[ST\\_SIP\\_MSG](#) \*pstMsg

The message.

ZUCHAR ucType

Type of the header.

[ST\\_ZOS\\_SSTR](#) \*pstStr

The string.

**Output parameters:**

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.2.8 Sip\_FillExtHdr

Fills an extension message header identified by its name with a string value.

```
ZINT Sip_FillExtHdr(ST\_SIP\_MSG *pstMsg, ZCHAR *pcName, ST\_ZOS\_SSTR *pstStr);
```

[Parameters]

**Input parameters:**

[ST\\_SIP\\_MSG](#) \*pstMsg

The message.

ZCHAR \*pcName

Name of the header.

[ST\\_ZOS\\_SSTR](#) \*pstStr

The string.

**Output parameters:**

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.2.9 Sip\_FindMsgHdr

Finds a message header by its type. ZNULL will be returned if the header has not been decoded. If the header has been found, the header address which is a union member of ST\_SIP\_MSG\_HDR will be returned.

```
ZVOID * Sip_FindMsgHdr(ST\_SIP\_MSG *pstMsg, ZUCHAR ucType);
```

[Parameters]

**Input parameters:**

[ST\\_SIP\\_MSG](#) \*pstMsg

The message.

ZUCHAR ucType

Type of the header.

**Output parameters:**

None.

**[Return value]**

Returns the address of the header on success, or ZNULL.

### 3.2.10 Sip\_FindMsgHdrX

Finds a message header by its type and index. ZNULL will be returned if the header has not been decoded. If the header has been found, the header address which is a union member of ST\_SIP\_MSG\_HDR will be returned.

```
ZVOID * Sip_FindMsgHdrX(ST\_SIP\_MSG *pstMsg, ZUCHAR ucType, ZINT iIndex);
```

**[Parameters]****Input parameters:**

[ST\\_SIP\\_MSG](#) \*pstMsg

The message.

ZUCHAR ucType

Type of the header.

ZINT iIndex

The index.

**Output parameters:**

None.

**[Return value]**

Returns the address of the header on success, or ZNULL.

### 3.2.11 Sip\_FindExtHdr

Finds an extension message header by its name.

```
ZINT Sip_FindExtHdr(ST\_SIP\_MSG *pstMsg, ZCHAR *pcName,  
ST\_SIP\_HDR\_EXT\_HDR **ppstHdr);
```

**[Parameters]****Input parameters:**

[ST\\_SIP\\_MSG](#) \*pstMsg

The message.

ZCHAR \*pcName

Name of the extension message header.

#### Output parameters:

[ST\\_SIP\\_HDR\\_EXT\\_HDR](#) \*\*ppstHdr

The address of the header on success, or ZNULL.

#### [Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.2.12 Sip\_FindExtHdrX

Finds an extension message header by its name and index.

```
ZINT Sip_FindExtHdrX(ST\_SIP\_MSG *pstMsg, ZCHAR *pcName, ZINT iIndex,  
                    ST\_SIP\_HDR\_EXT\_HDR **ppstHdr);
```

#### [Parameters]

##### Input parameters:

[ST\\_SIP\\_MSG](#) \*pstMsg

The message.

ZCHAR \*pcName

Name of the extension message header.

ZINT iIndex

The index.

##### Output parameters:

[ST\\_SIP\\_HDR\\_EXT\\_HDR](#) \*\*ppstHdr

The address of the header on success, or ZNULL.

#### [Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.2.13 Sip\_GetMimeBoundary

Gets the mime boundary from the content type.

```
ZINT Sip_GetMimeBoundary(ST SIP HDR CONTENT TYPE *pstContentType,
                        ST ZOS SSTR **ppstBoundary);
```

[Parameters]

**Input parameters:**

[ST SIP HDR CONTENT TYPE](#) \*pstContentType

The content type.

**Output parameters:**

[ST ZOS SSTR](#) \*\*ppstBoundary

The address of the mime boundary on success.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.2.14 Sip\_GetMimeBody

Gets the mime body string from the boundary.

```
ZINT Sip_GetMimeBody(ST ZOS SSTR *pstStr, ST ZOS SSTR *pstBoundary,
                    ST ZOS SSTR *pstBody);
```

[Parameters]

**Input parameters:**

[ST ZOS SSTR](#) \*pstStr

The string.

[ST ZOS SSTR](#) \*pstBoundary

The boundary.

[ST ZOS SSTR](#) \*pstBody

The mime body.

**Output parameters:**

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.2.15 Sip\_GetContentLen

Gets the content length from a SIP message.

```
ZINT Sip_GetContentLen(ST SIP MSG *pstMsg, ZULONG *pdwContentLen);
```

**[Parameters]****Input parameters:**

[ST SIP MSG](#) \*pstMsg

The message.

**Output parameters:**

ZULONG \*pdwContentLen

The content length.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

### 3.2.16 Sip\_GetContentLen2

Gets the content length from the string buffer.

```
ZINT Sip_GetContentLen2(ST ZOS SSTR *pstStr, ZULONG *pdwContentLen);
```

**[Parameters]****Input parameters:**

[ST ZOS SSTR](#) \*pstStr

The string.

**Output parameters:**

ZULONG \*pdwContentLen

The content length.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

### 3.2.17 Sip\_UpdateContentLen

Updates the content length in headers of a message.

```
ZINT Sip_UpdateContentLen(ST SIP MSG *pstMsg, ZULONG dwContentLen);
```

**[Parameters]****Input parameters:**

[ST SIP MSG](#) \*pstMsg

The message.

ZULONG dwContentLen

The new length.

**Output parameters:**

None.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

### 3.2.18 Sip\_GetViaBranch

Gets the via branch ID.

```
ST_ZOS_SSTR * Sip_GetViaBranch(ST_SIP_HDR_VIA *pstVia);
```

**[Parameters]****Input parameters:**

```
ST_SIP_HDR_VIA *pstVia
```

The via header.

**Output parameters:**

None.

**[Return value]**

Returns the branch ID, or ZNULL.

### 3.2.19 Sip\_HdrLstCreateHdr

Creates a message header and adds it into a header list.

```
ZVOID * Sip_HdrLstCreateHdr(ST_ZOS_DBUF *pstMemBuf ,  
ST_SIP_MSG_HDR_LST *pstHdrLst, ZUCHAR ucType);
```

**[Parameters]****Input parameters:**

```
ST_ZOS_DBUF *pstMemBuf
```

The memory buffer from which the memory for the header will be allocated.

```
ST_SIP_MSG_HDR_LST *pstHdrLst
```

The header list.

```
ZUCHAR ucType
```

Type of the header.

**Output parameters:**

None.

**[Return value]**

Returns the address of the header, or ZNULL.

### 3.2.20 Sip\_HdrLstDeleteHdr

Deletes the first header of a specific type from a header list.

```
ZINT Sip_HdrLstDeleteHdr(ST_SIP_MSG_HDR_LST *pstHdrLst, ZUCHAR ucType);
```

**[Parameters]****Input parameters:**

ST\_SIP\_MSG\_HDR\_LST \*pstHdrLst  
The header list.

ZUCHAR ucType  
Type of the header.

**Output parameters:**

None.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

### 3.2.21 Sip\_HdrLstFindHdr

Searches for a message header in a header list. ZNULL will be returned if the header has not been decoded, or its address which is the union member of ST\_SIP\_MSG\_HDR, will be returned.

```
ZVOID * Sip_HdrLstFindHdr(ST_SIP_MSG_HDR_LST *pstHdrLst, ZUCHAR ucType);
```

**[Parameters]****Input parameters:**

ST\_SIP\_MSG\_HDR\_LST \*pstHdrLst  
The header list.

ZUCHAR ucType  
Type of the header.

**Output parameters:**

None.

**[Return value]**

Returns the header address on success, or ZNULL.

### 3.2.22 Sip\_HdrLstGetContentLen

Gets the content length from a header list.

```
ZINT Sip_HdrLstGetContentLen(ST_SIP_MSG_HDR_LST *pstHdrLst,  
                             ZULONG *pdwContentLen);
```

[Parameters]

**Input parameters:**

ST\_SIP\_MSG\_HDR\_LST \*pstHdrLst  
The header list.

**Output parameters:**

ZULONG \*pdwContentLen  
The content length.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.2.23 Sip\_HdrLstUpdateContentLen

Updates the content length in a header list.

```
ZINT Sip_HdrLstUpdateContentLen(ST_ZOS_DBUF *pstMemBuf ,  
                                 ST_SIP_MSG_HDR_LST *pstHdrLst, ZULONG dwContentLen);
```

[Parameters]

**Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf  
The memory buffer.

ST\_SIP\_MSG\_HDR\_LST \*pstHdrLst  
The header list.

ZULONG dwContentLen  
The new content length.

**Output parameters:**

None.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

### 3.2.24 Sip\_MPartLstCreateMPart

Creates a multi-part and adds it to a multi-part list.

```
ZINT Sip_MPartLstCreateMPart(ST_ZOS_DBUF *pstMemBuf,  
                             ST_SIP_BODY_MPART_LST *pstMpartLst,  
                             ST_SIP_BODY_MPART **ppstMpart);
```

**[Parameters]****Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

ST\_SIP\_BODY\_MPART\_LST \*pstMpartLst

The multi-part list.

**Output parameters:**

[ST\\_SIP\\_BODY\\_MPART](#) \*\*ppstMpart

The newly created multi-part on success.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

### 3.2.25 Sip\_AbnfGetVersion

Gets the version of SIP ABNF.

```
ZCHAR * Sip_AbnfGetVersion();
```

**[Parameters]****Input parameters:**

None.

**Output parameters:**

None.

**[Return value]**

Returns the version string.

### 3.3 ABNF Config Interfaces

These interfaces are included in sip\_abnf\_cfg.h.

#### 3.3.1 Sip\_AbnfCfgGetLogLevel

Gets the log level.

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

[Parameters]

**Input parameters:**

None.

**Output parameters:**

ZULONG \*pdwLevel

The log level.

[Return value]

Returns ZOK.

#### 3.3.2 Sip\_AbnfCfgGetOption

Gets the config option. The option may be SIP\_ABNF\_OPT\_STRICT\_DECODE (strict decoding), SIP\_ABNF\_OPT\_SDP\_DECODE (SDP decoding), SIP\_ABNF\_OPT\_ADD\_CONTENT\_LEN (add content length).

```
ZINT Sip_AbnfCfgGetOption(ZULONG *pdwOption);
```

[Parameters]

**Input parameters:**

None.

**Output parameters:**

ZULONG \*pdwOption

The config option.

[Return value]

Returns ZOK.

#### 3.3.3 Sip\_AbnfCfgGetHdrDecode

Gets the a Boolean value indicating whether to decode a particular type of headers. The default is to decode the header.

```
ZINT Sip_AbnfCfgGetHdrDecode(ZUCHAR ucHdrType, ZBOOL *pbDecode);
```

[Parameters]

**Input parameters:**

ZUCHAR ucHdrType  
Type of the header.

**Output parameters:**

ZBOOL \*pbDecode  
The Boolean value.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.3.4 Sip\_AbnfCfgSetLogLevel

Sets the log level.

```
ZINT Sip_AbnfCfgSetLogLevel(ZULONG dwLevel);
```

[Parameters]

**Input parameters:**

ZULONG dwLevel  
The log level.

**Output parameters:**

None.

[Return value]

Returns ZOK.

### 3.3.5 Sip\_AbnfCfgSetOption

Sets the config option. The option may be SIP\_ABNF\_OPT\_STRICT\_DECODE (strict decoding), SIP\_ABNF\_OPT\_SDP\_DECODE (SDP decoding), SIP\_ABNF\_OPT\_ADD\_CONTENT\_LEN (add content length).

```
ZINT Sip_AbnfCfgSetOption(ZULONG dwOption);
```

[Parameters]

**Input parameters:**

ZULONG dwOption  
The config option.

**Output parameters:**

None.

**[Return value]**

Returns ZOK.

### 3.3.6 Sip\_AbnfCfgClrOption

Clear the config option. The option may be SIP\_ABNF\_OPT\_STRICT\_DECODE (strict decoding), SIP\_ABNF\_OPT\_SDP\_DECODE (SDP decoding), SIP\_ABNF\_OPT\_ADD\_CONTENT\_LEN (add content length).

```
ZINT Sip_AbnfCfgClrOption(ZULONG dwOption);
```

**[Parameters]****Input parameters:**

ZULONG dwOption

The option to clear.

**Output parameters:**

None.

**[Return value]**

Returns ZOK.

### 3.3.7 Sip\_AbnfCfgSetHdrDecode

Sets a Boolean value which indicates whether to decode a particular type of headers.

```
ZINT Sip_AbnfCfgSetHdrDecode(ZUCHAR ucHdrType, ZBOOL bDecode);
```

**[Parameters]****Input parameters:**

ZUCHAR ucHdrType

The header type.

**Output parameters:**

None.

**[Return value]**

Returns ZOK.

### 3.4 Primitive Interfaces

Here are some primitives invoked by the SIP user agent into the SIP stack to request a specific service. These interfaces are included in sip\_ui.h.

#### 3.4.1 Sip\_SendSimReq

[Description]

Send session initiating request, like INVITE.

[Declaration]

```
ZINT Sip_SendSimReq(ZULONG dwCompId, ZULONG dwSessUserId,
                    ZULONG dwDlgUserId, ZULONG dwTransUserId,
                    ST_SIP_TPT_ADDR *pstTptAddr, ST_SIP_MSG *pstMsg);
```

[Parameters]

<b>dwCompId</b>	The SIP service owner ID (component id).
<b>dwSessUserId</b>	The SIP session user ID.
<b>dwDlgUserId</b>	The SIP dialog user ID.
<b>dwTransUserId</b>	The SIP transaction user ID.
<b>pstTptAddr</b>	A pointer to SIP transport address.
<b>pstMsg</b>	A pointer to SIP message. It should include “request-line”, “from” and “to” at least.

[Return value]

<b>ZOK</b>	Request send to SIP stack successful.
<b>ZFAILED</b>	An error occurred.

For instructions on how to use this interface please see the [example](#).

#### 3.4.2 Sip\_SendSimRsp

[Description]

Send response to session initiating request, like 180 or 200 to INVITE.

[Declaration]

```
ZINT Sip_SendSimRsp(ZULONG dwCompId, ZULONG dwSessUserId,
                    ZULONG dwSessId, ZULONG dwDlgUserId, ZULONG dwDlgId,
                    ZULONG dwTransUserId, ZULONG dwTransId, ZULONG dwStatusCode,
                    ST_SIP_MSG *pstMsg);
```

[Parameters]

<b>dwCompId</b>	The SIP service owner ID (component id).
<b>dwSessUserId</b>	The SIP session user ID.
<b>dwDlgUserId</b>	The SIP dialog user ID.
<b>dwDlgId</b>	The SIP dialog ID

<b>dwTransUserId</b>	The SIP transaction user ID.
<b>dwTransId</b>	The SIP transaction ID.
<b>dwStatusCode</b>	The SIP Status-Code value.
<b>pstMsg</b>	A pointer to SIP message. It should include “request-line”, “from” and “to” at least.

[Return value]

<b>ZOK</b>	Request send to SIP stack successful.
<b>ZFAILED</b>	An error occurred.

For instructions on how to use this interface please see the [example](#).

### 3.4.3 Sip\_SendSsmReq

[Description]

Send session status request in a dialog, like PRACK, INFO, UPDATE, OPTION, REGISTER, MESSAGE, COMET, PUBLISH.

[Declaration]

```
ZINT Sip_SendSsmReq(ZULONG dwSessUserId, ZULONG dwSessId,
                   ZULONG dwDlgUserId, ZULONG dwDlgId, ZULONG dwTransUserId,
                   ST_SIP_TPT_ADDR *pstTptAddr, ZUCHAR ucMethodType,
                   ST_SIP_MSG *pstMsg);
```

[Parameters]

<b>dwSessUserId</b>	The SIP session user ID.
<b>dwSessId</b>	The SIP session ID.
<b>dwDlgUserId</b>	The SIP dialog user ID.
<b>dwDlgId</b>	The SIP dialog ID
<b>dwTransUserId</b>	The SIP transaction user ID.
<b>pstTptAddr</b>	A pointer to SIP transport address.
<b>ucMethodType</b>	SIP method type, defined in EN_SIP_METHOD.
<b>pstMsg</b>	A pointer to SIP message. It should include “request-line”, “from” and “to” at least.

[Return value]

<b>ZOK</b>	Request send to SIP stack successful.
<b>ZFAILED</b>	An error occurred.

For instructions on how to use this interface please see the [example](#).

### 3.4.4 Sip\_SendSsmRsp

[Description]

Send response to session status request in a dialog, like 200 to PRACK, INFO, UPDATE, OPTION, REGISTER, MESSAGE, COMET, PUBLISH.

[Declaration]

```
ZINT Sip_SendSsmRsp(ZULONG dwSessUserId, ZULONG dwSessId,
                    ZULONG dwDlgUserId, ZULONG dwDlgId, ZULONG dwTransUserId,
                    ZULONG dwTransId, ZUCHAR ucMethodType, ZULONG dwStatusCode,
                    ST_SIP_MSG *pstMsg);
```

[Parameters]

<b>dwSessUserId</b>	The SIP session user ID.
<b>dwSessId</b>	The SIP session ID.
<b>dwDlgUserId</b>	The SIP dialog user ID.
<b>dwDlgId</b>	The SIP dialog ID
<b>dwTransUserId</b>	The SIP transaction user ID.
<b>dwTransId</b>	The SIP transaction ID.
<b>ucMethodType</b>	SIP method type, defined in EN_SIP_METHOD.
<b>dwStatusCode</b>	The SIP Status-Code value.
<b>pstMsg</b>	A pointer to SIP message. It should include “request-line”, “from” and “to” at least.

[Return value]

<b>ZOK</b>	Request send to SIP stack successful.
<b>ZFAILED</b>	An error occurred.

For instructions on how to use this interface please see the [example](#).

### 3.4.5 Sip\_SendSamReq

[Description]

Send session ACK request, like ACK to 200 or 3xx.

[Declaration]

```
ZINT Sip_SendSamReq(ZULONG dwSessUserId, ZULONG dwSessId,
                    ZULONG dwDlgUserId, ZULONG dwDlgId, ZULONG dwTransUserId,
                    ST_SIP_TPT_ADDR *pstTptAddr, ST_SIP_MSG *pstMsg);
```

[Parameters]

<b>dwSessUserId</b>	The SIP session user ID.
<b>dwSessId</b>	The SIP session ID.
<b>dwDlgUserId</b>	The SIP dialog user ID.
<b>dwDlgId</b>	The SIP dialog ID
<b>dwTransUserId</b>	The SIP transaction user ID.
<b>pstTptAddr</b>	A pointer to SIP transport address.

**pstMsg** A pointer to SIP message. It should include “request-line”, “from” and “to” at least.

[Return value]

**ZOK** Request send to SIP stack successful.

**ZFAILED** An error occurred.

For instructions on how to use this interface please see the [example](#).

### 3.4.6 Sip\_SendScmReq

[Description]

Send session CANCEL request, like CANCEL after 1xx received.

[Declaration]

```
ZINT Sip_SendScmReq(ZULONG dwSessUserId, ZULONG dwSessId,
                    ZULONG dwDlgUserId, ZULONG dwDlgId, ZULONG dwTransUserId,
                    ST_SIP_TPT_ADDR *pstTptAddr, ST_SIP_MSG *pstMsg);
```

[Parameters]

**dwSessUserId** The SIP session user ID.

**dwSessId** The SIP session ID.

**dwDlgUserId** The SIP dialog user ID.

**dwDlgId** The SIP dialog ID

**dwTransUserId** The SIP transaction user ID.

**pstTptAddr** A pointer to SIP transport address.

**pstMsg** A pointer to SIP message. It should include “request-line”, “from” and “to” at least.

[Return value]

**ZOK** Request send to SIP stack successful.

**ZFAILED** An error occurred.

For instructions on how to use this interface please see the [example](#).

### 3.4.7 Sip\_SendSmmReq

[Description]

Send session modify request, like re-INVITE or UPDATE.

[Declaration]

```
ZINT Sip_SendSmmReq(ZULONG dwSessUserId, ZULONG dwSessId,
                   ZULONG dwDlgUserId, ZULONG dwDlgId, ZULONG dwTransUserId,
                   ST_SIP_TPT_ADDR *pstTptAddr, ST_SIP_MSG *pstMsg);
```

## [Parameters]

<b>dwSessUserId</b>	The SIP session user ID.
<b>dwSessId</b>	The SIP session ID.
<b>dwDlgUserId</b>	The SIP dialog user ID.
<b>dwDlgId</b>	The SIP dialog ID
<b>dwTransUserId</b>	The SIP transaction user ID.
<b>pstTptAddr</b>	A pointer to SIP transport address.
<b>pstMsg</b>	A pointer to SIP message. It should include “request-line”, “from” and “to” at least.

## [Return value]

<b>ZOK</b>	Request send to SIP stack successful.
<b>ZFAILED</b>	An error occurred.

For instructions on how to use this interface please see the [example](#).

### 3.4.8 Sip\_SendSmmRsp

## [Description]

Send response to session modify request, like 200 to re-INVITE or UPDATE.

## [Declaration]

```
ZINT Sip_SendSmmRsp(ZULONG dwSessUserId, ZULONG dwSessId,
                   ZULONG dwDlgUserId, ZULONG dwDlgId, ZULONG dwTransUserId,
                   ZULONG dwTransId, ZULONG dwStatusCode, ST_SIP_MSG *pstMsg);
```

## [Parameters]

<b>dwSessUserId</b>	The SIP session user ID.
<b>dwSessId</b>	The SIP session ID.
<b>dwDlgUserId</b>	The SIP dialog user ID.
<b>dwDlgId</b>	The SIP dialog ID
<b>dwTransUserId</b>	The SIP transaction user ID.
<b>dwTransId</b>	The SIP transaction ID.
<b>dwStatusCode</b>	The SIP Status-Code value.
<b>pstMsg</b>	A pointer to SIP message. It should include “request-line”, “from” and “to” at least.

## [Return value]

<b>ZOK</b>	Request send to SIP stack successful.
<b>ZFAILED</b>	An error occurred.

For instructions on how to use this interface please see the [example](#).

### 3.4.9 Sip\_SendStmReq

[Description]

Send session terminate request, like BYE.

[Declaration]

```
ZINT Sip_SendStmReq(ZULONG dwSessUserId, ZULONG dwSessId,
                    ZULONG dwDlgUserId, ZULONG dwDlgId, ZULONG dwTransUserId,
                    ST_SIP_TPT_ADDR *pstTptAddr, ST_SIP_MSG *pstMsg);
```

[Parameters]

<b>dwSessUserId</b>	The SIP session user ID.
<b>dwSessId</b>	The SIP session ID.
<b>dwDlgUserId</b>	The SIP dialog user ID.
<b>dwDlgId</b>	The SIP dialog ID
<b>dwTransUserId</b>	The SIP transaction user ID.
<b>pstTptAddr</b>	A pointer to SIP transport address.
<b>pstMsg</b>	A pointer to SIP message. It should include “request-line”, “from” and “to” at least.

[Return value]

<b>ZOK</b>	Request send to SIP stack successful.
<b>ZFAILED</b>	An error occurred.

For instructions on how to use this interface please see the [example](#).

### 3.4.10 Sip\_SendDamReq

[Description]

Send dialog associate message request, like SUBSCRIBE, REFER, NOTIFY.

[Declaration]

```
ZINT Sip_SendDamReq(ZULONG dwCompId, ZULONG dwSessUserId,
                    ZULONG dwSessId, ZULONG dwDlgUserId, ZULONG dwDlgId,
                    ZULONG dwTransUserId, ST_SIP_TPT_ADDR *pstTptAddr,
                    ZUCHAR ucMethodType, ST_SIP_MSG *pstMsg);
```

[Parameters]

<b>dwCompId</b>	The SIP service owner ID (component id).
-----------------	--

<b>dwSessUserId</b>	The SIP session user ID.
<b>dwSessId</b>	The SIP session ID.
<b>dwDlgUserId</b>	The SIP dialog user ID.
<b>dwDlgId</b>	The SIP dialog ID
<b>dwTransUserId</b>	The SIP transaction user ID.
<b>pstTptAddr</b>	A pointer to SIP transport address.
<b>ucMethodType</b>	SIP method type, defined in EN_SIP_METHOD.
<b>pstMsg</b>	A pointer to SIP message. It should include “request-line”, “from” and “to” at least.

[Return value]

<b>ZOK</b>	Request send to SIP stack successful.
<b>ZFAILED</b>	An error occurred.

For instructions on how to use this interface please see the [example](#).

### 3.4.11 Sip\_SendDamReqX

[Description]

Send dialog associate message request, only for REFER.

[Declaration]

```
ZINT Sip_SendDamReqX(ZULONG dwCompId, ZULONG dwSessUserId,
                    ZULONG dwSessId, ZULONG dwDlgUserId, ZULONG dwDlgId,
                    ZULONG dwTransUserId, ST_SIP_TPT_ADDR *pstTptAddr,
                    ZULONG dwReferExpires, ST_SIP_MSG *pstMsg);
```

[Parameters]

<b>dwCompId</b>	The SIP service owner ID (component id).
<b>dwSessUserId</b>	The SIP session user ID.
<b>dwSessId</b>	The SIP session ID.
<b>dwDlgUserId</b>	The SIP dialog user ID.
<b>dwDlgId</b>	The SIP dialog ID
<b>dwTransUserId</b>	The SIP transaction user ID.
<b>pstTptAddr</b>	A pointer to SIP transport address.
<b>dwReferExpires</b>	The expires value for REFER method.
<b>pstMsg</b>	A pointer to SIP message. It should include “request-line”, “from” and “to” at least.

[Return value]

<b>ZOK</b>	Request send to SIP stack successful.
<b>ZFAILED</b>	An error occurred.

### 3.4.12 Sip\_SendDamRsp

[Description]

Send response to dialog associate message request, like 200 to SUBSCRIBE, REFER, NOTIFY.

[Declaration]

```
ZINT Sip_SendDamRsp(ZULONG dwCompId, ZULONG dwSessUserId,
                    ZULONG dwSessId, ZULONG dwDlgUserId, ZULONG dwDlgId,
                    ZULONG dwTransUserId, ZULONG dwTransId, ZUCHAR ucMethodType,
                    ZULONG dwStatusCode, ST_SIP_MSG *pstMsg);
```

[Parameters]

<b>dwCompId</b>	The SIP service owner ID (component id).
<b>dwSessUserId</b>	The SIP session user ID.
<b>dwSessId</b>	The SIP session ID.
<b>dwDlgUserId</b>	The SIP dialog user ID.
<b>dwDlgId</b>	The SIP dialog ID
<b>dwTransUserId</b>	The SIP transaction user ID.
<b>dwTransId</b>	The SIP transaction ID.
<b>ucMethodType</b>	SIP method type, defined in EN_SIP_METHOD.
<b>dwStatusCode</b>	The SIP Status-Code value.
<b>pstMsg</b>	A pointer to SIP message. It should include “request-line”, “from” and “to” at least.

[Return value]

<b>ZOK</b>	Request send to SIP stack successful.
<b>ZFAILED</b>	An error occurred.

### 3.4.13 Sip\_SendCimReq

[Description]

Send call independent message request, like OPTIONS, REGISTER, MESSAGE, COMET, PUBLISH.

[Declaration]

```
ZINT Sip_SendCimReq(ZULONG dwCompId, ZULONG dwSessUserId,
                    ZULONG dwDlgUserId, ZULONG dwTransUserId,
                    ST_SIP_TPT_ADDR *pstTptAddr, ZUCHAR ucMethodType,
                    ST_SIP_MSG *pstMsg);
```

[Parameters]

<b>dwCompId</b>	The SIP service owner ID (component id).
<b>dwSessUserId</b>	The SIP session user ID.
<b>dwDlgUserId</b>	The SIP dialog user ID.
<b>dwTransUserId</b>	The SIP transaction user ID.
<b>pstTptAddr</b>	A pointer to SIP transport address.

**ucMethodType** SIP method type, defined in EN\_SIP\_METHOD.  
**pstMsg** A pointer to SIP message. It should include “request-line”, “from” and “to” at least.

[Return value]

**ZOK** Request send to SIP stack successful.

**ZFAILED** An error occurred.

For instructions on how to use this interface please see the [example](#).

### 3.4.14 Sip\_SendCimRsp

[Description]

Send response to call independent message request, like 200 to OPTIONS, REGISTER, MESSAGE, COMET, PUBLISH.

[Declaration]

```
ZINT Sip_SendCimRsp(ZULONG dwCompId, ZULONG dwSessUserId,
                   ZULONG dwSessId, ZULONG dwDlgUserId, ZULONG dwDlgId,
                   ZULONG dwTransUserId, ZULONG dwTransId, ZUCHAR ucMethodType,
                   ZULONG dwStatusCode, ST_SIP_MSG *pstMsg);
```

[Parameters]

**dwCompId** The SIP service owner ID (component id).

**dwSessUserId** The SIP session user ID.

**dwSessId** The SIP session ID.

**dwDlgUserId** The SIP dialog user ID.

**dwDlgId** The SIP dialog ID.

**dwTransUserId** The SIP transaction user ID.

**dwTransId** The SIP transaction ID.

**ucMethodType** SIP method type, defined in EN\_SIP\_METHOD.

**dwStatusCode** The SIP Status-Code value.

**pstMsg** A pointer to SIP message. It should include “request-line”, “from” and “to” at least.

[Return value]

**ZOK** Request send to SIP stack successful.

**ZFAILED** An error occurred.

For instructions on how to use this interface please see the [example](#).

### 3.4.15 Indication and Confirmation Messages

These messages are invoked by the SIP stack to indicate the reception of a request (such as an INVITE indication) from another SIP Stack of a remote system. Then the indication and confirmation messages sent by SIP stack are processed by the SUA.

These messages will be sent to SUA directly while SIP stack received SIP messages from network. These message types are included in the types below. See [EN\\_SIP\\_SESS\\_EVNT\\_TYPE](#) for more details.

<b>EN_SIP_SESSE_ERR_IND</b>	A error occurred in SIP stack.
<b>EN_SIP_SESSE_SIM_CNF</b>	SIP stack received INVITE response.
<b>EN_SIP_SESSE_SIM_IND</b>	SIP stack received INVITE request.
<b>EN_SIP_SESSE_SSM_CNF</b>	SIP stack received Re-INVITE response.
<b>EN_SIP_SESSE_SSM_IND</b>	SIP stack received Re-INVITE request.
<b>EN_SIP_SESSE_SAM_IND</b>	SIP stack received ACK request.
<b>EN_SIP_SESSE_SCM_IND</b>	SIP stack received CANCEL request.
<b>EN_SIP_SESSE_SMM_CNF</b>	SIP stack received response to PRACK, INFO, UPDATE, OPTION, REGISTER, MESSAGE, COMET, PUBLISH.
<b>EN_SIP_SESSE_SMM_IND</b>	SIP stack received request of PRACK, INFO, UPDATE, OPTION, REGISTER, MESSAGE, COMET, PUBLISH.
<b>EN_SIP_SESSE_STM_CNF</b>	SIP stack received BYE response.
<b>EN_SIP_SESSE_STM_IND</b>	SIP stack received BYE request.
<b>EN_SIP_SESSE_DAM_CNF</b>	SIP stack received response to SUBSCRIBE, REFER, NOTIFY
<b>EN_SIP_SESSE_DAM_IND</b>	SIP stack received request of SUBSCRIBE, REFER, NOTIFY
<b>EN_SIP_SESSE_CIM_CNF</b>	SIP stack received response to OPTIONS, REGISTER, MESSAGE, COMET, PUBLISH, which is not in a exist dialog.
<b>EN_SIP_SESSE_CIM_IND</b>	SIP stack received request of OPTIONS, REGISTER, MESSAGE, COMET, PUBLISH, which is not in a exist dialog.

Then how does SIP know to which task these messages should sent? This task should be designated by the user by invoking function [Sip\\_CfgSetUpperTaskId](#). The user needs to provide the task ID as the input parameter. The task ID for SUA upper on the SIP Stack is ZTASKID\_SUA.

In order to process these indication and confirmation messages, SUA should implement the function which is responsible for handling these messages. The function pointer has to be provided to SUA's task. This shall be done as SUA starts. Then when SIP stack sends the messages to SUA's task, and the function will be processed at the SUA's task immediately.

For example, when SIP stack has received an INVITE response to the INVITE request that has been sent before, such as 200 OK response (as is shown in [figure 4-2](#)), SIP stack task will invoke `Sip_SendSimCnf` to send a message with an event type `EN_SIP_SESSE_SIM_CNF` to SUA's task and the message will be processed immediately by the function mentioned aboved.

Suppose the function is named `MessageProcess`.

```

/* zos message */
typedef struct tagZOS_MSG
{
    ST\_ZOS\_MSP stMsp;           /* module service post */
    ZULONG dwMsgLen;          /* message length */
    ZULONG dwDataLen;        /* data length beside message header */
} ST_ZOS_MSG;

/* Process messages sent by SIP. */
MessageProcess(ST_ZOS_MSG *pstMsg)
{
    .....
}

/* Let SUA know function MessageProcess that is responsible for handling indication
and confirmation messages when it starts. */
Sua_Start()
{
    .....
    /* module start */
    if (ZOS_MOD_START(ZTASKID_SUA, ZTASK_PRIORITY_NORMAL, 0,
                     MessageProcess) != ZOK)
        return ZFAILED;

    return ZOK;
}

```

Users have to let the SIP stack know to which task the indication and confirmation messages should be sent.

## 3.5 Message Interfaces

### 3.5.1 Sip\_MsgCreate

Creates a SIP message.

```
ZINT Sip_MsgCreate(ST SIP MSG **ppstMsg);
```

[Parameters]

#### Input parameters:

None.

#### Output parameters:

[ST\\_SIP\\_MSG](#) \*\*ppstMsg

The newly created SIP message on success.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.2 Sip\_MsgCreateX

Creates a SIP message with a particular memory buffer.

```
ZINT Sip_MsgCreateX(ST\_ZOS\_DBUF *pstMemBuf, ST\_SIP\_MSG **ppstMsg);
```

[Parameters]

**Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

**Output parameters:**

[ST\\_SIP\\_MSG](#) \*\*ppstMsg

The newly created SIP message on success.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.3 Sip\_MsgDelete

Deletes a SIP message.

```
ZINT Sip_MsgDelete(ST\_SIP\_MSG *pstMsg);
```

[Parameters]

**Input parameters:**

[ST\\_SIP\\_MSG](#) \*pstMsg

The SIP message to delete.

**Output parameters:**

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.4 Sip\_MsgFillReqLineByIp

Fills a request line by the IP address.

```
ZINT Sip_MsgFillReqLineByIp(ST_ZOS_DBUF *pstMemBuf,
                             ST_SIP_MSG *pstMsg, ZUCHAR ucMethod, ST_ZOS_SSTR *pstUserInfo,
                             ST_ZOS_INET_ADDR *pstAddr);
```

[Parameters]

#### Input parameters:

ST\_ZOS\_DBUF \*pstMemBuf

The memory buffer.

ST\_SIP\_MSG \*pstMsg

The message whose request line is to be filled.

ZUCHAR ucMethod

The method.

ST\_ZOS\_SSTR \*pstUserInfo

The user information.

ST\_ZOS\_INET\_ADDR \*pstAddr

The address.

#### Output parameters:

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.5 Sip\_MsgFillReqLineByName

Fills a request line by the host name.

```
ZINT Sip_MsgFillReqLineByName(ST_ZOS_DBUF *pstMemBuf,
                               ST_SIP_MSG *pstMsg, ZUCHAR ucMethod, ST_ZOS_SSTR *pstUserInfo,
                               ST_ZOS_SSTR *pstHostname, ZULONG dwPort);
```

[Parameters]

#### Input parameters:

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_SIP\\_MSG](#) \*pstMsg

The message whose request line is to be filled.

ZUCHAR ucMethod

The method.

[ST\\_ZOS\\_SSTR](#) \*pstUserInfo

The user information.

[ST\\_ZOS\\_SSTR](#) \*pstHostname

The host name.

ZULONG dwPort

The port.

#### **Output parameters:**

None.

#### **[Return value]**

Returns ZOK on success, or ZFAILED on failure.

### **3.5.6 Sip\_MsgFillReqLineBySipUri**

Fills a request line by the SIP URI.

```
ZINT Sip_MsgFillReqLineBySipUri(ST\_ZOS\_DBUF *pstMemBuf,  
                                ST\_SIP\_MSG *pstMsg, ZUCHAR ucMethod, ST\_SIP\_SIP\_URI *pstSipUri);
```

#### **[Parameters]**

#### **Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_SIP\\_MSG](#) \*pstMsg

The message whose request line is to be filled.

ZUCHAR ucMethod

The method.

[ST\\_SIP\\_SIP\\_URI](#) \*pstSipUri

The SIP URI.

#### Output parameters:

None.

#### [Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.7 Sip\_MsgFillReqLineByTelUri

Fills a request line by the tel URI.

```
ZINT Sip_MsgFillReqLineByTelUri(ST\_ZOS\_DBUF *pstMemBuf,  
                                ST\_SIP\_MSG *pstMsg, ZUCHAR ucMethod, ST\_SIP\_TEL\_URI *pstTelUri);
```

#### [Parameters]

##### Input parameters:

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_SIP\\_MSG](#) \*pstMsg

The message whose request line is to be filled.

ZUCHAR ucMethod

The method.

[ST\\_SIP\\_TEL\\_URI](#) \*pstTelUri

The tele URI.

#### Output parameters:

None.

#### [Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.8 Sip\_MsgFillStatusLine

Fills a message's status line.

```
ZINT Sip_MsgFillStatusLine(ST SIP MSG *pstMsg, ZULONG dwStatusCode);
```

[Parameters]

**Input parameters:**

[ST SIP MSG](#) \*pstMsg

The message.

ZULONG dwStatusCode

The status code.

**Output parameters:**

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.9 Sip\_MsgFillHdrCallId

Fills a message's Call-ID.

```
ZINT Sip_MsgFillHdrCallId(ST SIP MSG *pstMsg, ST SIP CALLID *pstCallId);
```

[Parameters]

**Input parameters:**

[ST SIP MSG](#) \*pstMsg

The message.

[ST SIP CALLID](#) \*pstCallId

The call ID.

**Output parameters:**

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.10 Sip\_MsgFillHdrCSeq

Fills a message's CSeq header.

```
ZINT Sip_MsgFillHdrCSeq(ST SIP MSG *pstMsg, ZUCHAR ucMethod,  
                        ZULONG dwSeq);
```

[Parameters]

**Input parameters:**

[ST SIP MSG](#) \*pstMsg

The message.

ZUCHAR ucMethod

The method.

ZULONG dwSeq

The CSeq value.

**Output parameters:**

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.11 Sip\_MsgFillHdrSupted

Fills a message's supported header.

```
ZINT Sip_MsgFillHdrSupted(ST SIP MSG *pstMsg, ZULONG dwSuptFlag);
```

[Parameters]

**Input parameters:**

[ST SIP MSG](#) \*pstMsg

The message.

ZULONG dwSuptFlag

The supporting flag.

**Output parameters:**

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.12 Sip\_MsgFillHdrAllow

Fills a message's Allow header.

```
ZINT Sip_MsgFillHdrAllow(ST SIP MSG *pstMsg, ZULONG dwSuptFlag);
```

[Parameters]

**Input parameters:**

[ST SIP MSG](#) \*pstMsg

The message.

ZULONG dwSuptFlag

The flag.

**Output parameters:**

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.13 Sip\_MsgFillHdrRequire

Fills a message's require header.

```
ZINT Sip_MsgFillHdrRequire(ST SIP MSG *pstMsg, ZULONG dwSuptFlag);
```

[Parameters]

**Input parameters:**

[ST SIP MSG](#) \*pstMsg

The message.

ZULONG dwSuptFlag

The requiring flag.

**Output parameters:**

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.14 Sip\_MsgFillHdrSessExpire

Fills a message's Session-Expires header.

```
ZINT Sip_MsgFillHdrSessExpire(ST\_SIP\_MSG *pstMsg, ZBOOL bIsRefresher,  
                               ZBOOL bIsUasRefresher, ZULONG dwExpire);
```

[Parameters]

**Input parameters:**

[ST\\_SIP\\_MSG](#) \*pstMsg

The message.

ZBOOL bIsRefresher

The refresher present flag.

ZBOOL bIsUasRefresher

The refresher uas present flag.

ZULONG dwExpire

The delta-seconds.

**Output parameters:**

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.15 Sip\_MsgFillHdrMinSe

Fills a message's Min-Se header.

```
ZINT Sip_MsgFillHdrMinSe(ST\_SIP\_MSG *pstMsg, ZULONG dwExpire);
```

[Parameters]

**Input parameters:**

[ST\\_SIP\\_MSG](#) \*pstMsg

The message.

ZULONG dwExpire

The delta-seconds.

**Output parameters:**

None.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

### 3.5.16 Sip\_MsgFillHdrEvt

Fills a message's event header.

```
ZINT Sip_MsgFillHdrEvt(ST SIP MSG *pstMsg, ZUCHAR ucPkgType,  
                      ZUCHAR ucTempType);
```

**[Parameters]****Input parameters:**

[ST SIP MSG](#) \*pstMsg

The message.

ZUCHAR ucPkgType

The package type.

ZUCHAR ucTempType

The event template type.

**Output parameters:**

None.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

### 3.5.17 Sip\_MsgFillHdrExpire

Fills a message's Expire header.

```
ZINT Sip_MsgFillHdrExpire(ST SIP MSG *pstMsg, ZULONG dwExpire);
```

**[Parameters]****Input parameters:**

[ST SIP MSG](#) \*pstMsg

The message.

ZULONG dwExpire

The expire value.

**Output parameters:**

None.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

### 3.5.18 Sip\_MsgFillHdrSubsSta

Creates a subscription state header for a message and sets the state.

```
ZINT Sip_MsgFillHdrSubsSta(ST SIP MSG *pstMsg, ZUCHAR ucState);
```

**[Parameters]****Input parameters:**

[ST SIP MSG](#) \*pstMsg

The message.

ZUCHAR ucState

The state.

**Output parameters:**

None.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

### 3.5.19 Sip\_MsgFillHdrRAck

Fills a message's RAck header.

```
ZINT Sip_MsgFillHdrRAck(ST SIP MSG *pstMsg, ZULONG dwCseqNum,  
                        ZULONG dwRspNum, ZUCHAR ucMethod);
```

**[Parameters]****Input parameters:**

[ST SIP MSG](#) \*pstMsg

The message.

ZULONG dwRspNum

Response number.

ZULONG dwCseqNum

Cseq number.

ZUCHAR ucMethod

The method.

**Output parameters:**

None.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

### 3.5.20 Sip\_MsgFillHdrPrivacy

Creates a Privacy header for a message and sets the header.

```
ZINT Sip_MsgFillHdrPrivacy(ST\_SIP\_MSG *pstMsg, ZUCHAR ucPrivVal);
```

**[Parameters]****Input parameters:**

[ST\\_SIP\\_MSG](#) \*pstMsg

The message.

ZUCHAR ucPrivVal

Value of the header to create.

**Output parameters:**

None.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

### 3.5.21 Sip\_MsgGetFromToTag

Gets a From tag or To tag in a SIP message.

```
ZINT Sip_MsgGetFromToTag(ST\_SIP\_MSG *pstMsg, ZBOOL bFromTag,  
                          ST\_ZOS\_SSTR **ppstFromToTag);
```

**[Parameters]****Input parameters:**

[ST\\_SIP\\_MSG](#) \*pstMsg

The SIP message.

ZBOOL bFromTag

If it is ZTRUE the From tag will be returned, or the To tag.

**Output parameters:**

[ST\\_ZOS\\_SSTR](#) \*\*ppstFromToTag

The From or To tag.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

### 3.5.22 Sip\_HdrFillCseq

Fills the CSeq header.

```
ZINT Sip_HdrFillCseq(ST\_SIP\_HDR\_CSEQ *pstCseq, ZULONG dwCseqVal,
                    ZUCHAR ucMethod);
```

**[Parameters]**

**Input parameters:**

ZULONG dwCseqVal  
CSeq value.

ZUCHAR ucMethod  
The method.

**Output parameters:**

[ST\\_SIP\\_HDR\\_CSEQ](#) \*pstCseq  
The CSeq header.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

### 3.5.23 Sip\_HdrFillFromToByIp

Fills the From or To header by the IP address.

```
ZINT Sip_HdrFillFromToByIp(ST\_ZOS\_DBUF *pstMemBuf,
                           ST\_SIP\_HDR\_FROM\_TO *pstFromTo, ST\_ZOS\_SSTR *pstUserName,
                           ST\_ZOS\_SSTR *pstUserInfo, ST\_ZOS\_INET\_ADDR *pstAddr,
                           ST\_ZOS\_SSTR *pstTag)
```

**[Parameters]**

**Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_ZOS\\_SSTR](#) \*pstUserName

User name.

[ST\\_ZOS\\_SSTR](#) \*pstUserInfo

User information.

[ST\\_ZOS\\_INET\\_ADDR](#) \*pstAddr

The IP address.

[ST\\_ZOS\\_SSTR](#) \*pstTag

The tag.

### Output parameters:

[ST\\_SIP\\_HDR\\_FROM\\_TO](#) \*pstFromTo

The From or To header.

### [Return value]

Returns ZOK on success, or ZFAILED on failure.

## 3.5.24 Sip\_HdrFillFromToByName

Fills the From or To header by the host name.

```
ZINT Sip_HdrFillFromToByName(ST\_ZOS\_DBUF *pstMemBuf,  
                             ST\_SIP\_HDR\_FROM\_TO *pstFromTo, ST\_ZOS\_SSTR *pstUserName,  
                             ST\_ZOS\_SSTR *pstUserInfo, ST\_ZOS\_SSTR *pstHostname,  
                             ZULONG dwPort, ST\_ZOS\_SSTR *pstTag);
```

### [Parameters]

#### Input parameters:

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_ZOS\\_SSTR](#) \*pstUserName

User name.

[ST\\_ZOS\\_SSTR](#) \*pstUserInfo

User information.

[ST\\_ZOS\\_SSTR](#) \*pstHostname

The host name.

ZULONG dwPort

Port number.

[ST\\_ZOS\\_SSTR](#) \*pstTag

The tag.

#### Output parameters:

[ST\\_SIP\\_HDR\\_FROM\\_TO](#) \*pstFromTo

The From or To header.

#### [Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.25 Sip\_HdrFillFromToBySipUri

Fills the From or To header by the SIP URI.

```
ZINT Sip_HdrFillFromToBySipUri(ST\_ZOS\_DBUF *pstMemBuf,  
                               ST\_SIP\_HDR\_FROM\_TO *pstFromTo, ST\_ZOS\_SSTR *pstUserName,  
                               ST\_SIP\_SIP\_URI *pstSipUri, ST\_ZOS\_SSTR *pstTag);
```

#### [Parameters]

#### Input parameters:

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_ZOS\\_SSTR](#) \*pstUserName

User name.

[ST\\_SIP\\_SIP\\_URI](#) \*pstSipUri

The SIP URI.

[ST\\_ZOS\\_SSTR](#) \*pstTag

The tag.

#### Output parameters:

[ST\\_SIP\\_HDR\\_FROM\\_TO](#) \*pstFromTo

The From or To header.

#### [Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.26 Sip\_HdrFillFromToBySipsUri

Fills the From or To header by the SIPS URI.

```
ZINT Sip_HdrFillFromToBySipsUri(ST\_ZOS\_DBUF *pstMemBuf,  
                                ST\_SIP\_HDR\_FROM\_TO *pstFromTo, ST\_ZOS\_SSTR *pstUserName,  
                                ST\_SIP\_SIP\_URI *pstSipsUri, ST\_ZOS\_SSTR *pstTag);
```

#### [Parameters]

##### Input parameters:

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_ZOS\\_SSTR](#) \*pstUserName

The user name.

[ST\\_SIP\\_SIP\\_URI](#) \*pstSipsUri

The SIPS URI.

[ST\\_ZOS\\_SSTR](#) \*pstTag

The tag.

##### Output parameters:

[ST SIP HDR FROM TO](#) \*pstFromTo

The From or To header.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.27 Sip\_HdrFillFromToByTelUri

Fills the From or To header by the tel URI.

```
ZINT Sip_HdrFillFromToByTelUri(ST ZOS DBUF *pstMemBuf,
                               ST SIP HDR FROM TO *pstFromTo, ST ZOS SSTR *pstUserName,
                               ST SIP TEL URI *pstTelUri, ST ZOS SSTR *pstTag);
```

[Parameters]

**Input parameters:**

[ST ZOS DBUF](#) \*pstMemBuf

The memory buffer.

[ST ZOS SSTR](#) \*pstUserName

The user name.

[ST SIP TEL URI](#) \*pstTelUri

The tel URI.

[ST ZOS SSTR](#) \*pstTag

The tag.

**Output parameters:**

[ST SIP HDR FROM TO](#) \*pstFromTo

The From or To header.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.28 Sip\_HdrFillFromToByImUri

Fills the From or To header by the Instant Message (IM) URI.

```
ZINT Sip_HdrFillFromToByImUri(ST ZOS DBUF *pstMemBuf,
                               ST SIP HDR FROM TO *pstFromTo, ST ZOS SSTR *pstUserName,
                               ST SIP IM URI *pstImUri, ST ZOS SSTR *pstTag);
```

[Parameters]

**Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_ZOS\\_SSTR](#) \*pstUserName

The user name.

[ST\\_SIP\\_IM\\_URI](#) \*pstImUri

The IM URI.

[ST\\_ZOS\\_SSTR](#) \*pstTag

The tag.

**Output parameters:**

[ST\\_SIP\\_HDR\\_FROM\\_TO](#) \*pstFromTo

The From or To header.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

### 3.5.29 Sip\_HdrFillReferToByIp

Fills the refer-to header by the IP address.

```
ZINT Sip_HdrFillReferToByIp(ST\_ZOS\_DBUF *pstMemBuf,  
                            ST\_SIP\_HDR\_REFERER\_TO *pstReferTo, ST\_ZOS\_SSTR *pstUserName,  
                            ST\_ZOS\_SSTR *pstUserInfo, ST\_ZOS\_INET\_ADDR *pstAddr);
```

**[Parameters]****Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_ZOS\\_SSTR](#) \*pstUserName

The user name.

[ST\\_ZOS\\_SSTR](#) \*pstUserInfo

User information.

[ST\\_ZOS\\_INET\\_ADDR](#) \*pstAddr

The IP address.

**Output parameters:**

[ST SIP HDR REFER TO](#) \*pstReferTo

The refer-to header.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.30 Sip\_HdrFillReferToByName

Fills the refer-to header by the host name.

```
ZINT Sip_HdrFillReferToByName(ST ZOS DBUF *pstMemBuf,
                               ST SIP HDR REFER TO *pstReferTo, ST ZOS SSTR *pstUserName,
                               ST ZOS SSTR *pstUserInfo, ST ZOS SSTR *pstHostname,
                               ZULONG dwPort);
```

[Parameters]

**Input parameters:**

[ST ZOS DBUF](#) \*pstMemBuf

The memory buffer.

[ST ZOS SSTR](#) \*pstUserName

The user name.

[ST ZOS SSTR](#) \*pstUserInfo

User information.

[ST ZOS SSTR](#) \*pstHostname

The host name.

ZULONG dwPort

The port number.

**Output parameters:**

[ST SIP HDR REFER TO](#) \*pstReferTo

The refer-to header.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.31 Sip\_HdrFillReferredByByIp

Fills the referred-by header by the IP address.

```
ZINT Sip_HdrFillReferredByByIp(ST_ZOS_DBUF *pstMemBuf,  
                               ST_SIP_HDR_REFERRED_BY *pstReferredBy, ST_ZOS_SSTR *pstUserName,  
                               ST_ZOS_SSTR *pstUserInfo, ST_ZOS_INET_ADDR *pstAddr);
```

[Parameters]

**Input parameters:**

ST\_ZOS\_DBUF \*pstMemBuf

The memory buffer.

ST\_ZOS\_SSTR \*pstUserName

The user name.

ST\_ZOS\_SSTR \*pstUserInfo

User information.

ST\_ZOS\_INET\_ADDR \*pstAddr

The IP address.

**Output parameters:**

ST\_SIP\_HDR\_REFERRED\_BY \*pstReferredBy

The referred-by header.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.32 Sip\_HdrFillReferredByByName

Fills the referred-by header by the host name.

```
ZINT Sip_HdrFillReferredByByName(ST_ZOS_DBUF *pstMemBuf,  
                                 ST_SIP_HDR_REFERRED_BY *pstReferredBy, ST_ZOS_SSTR *pstUserName,  
                                 ST_ZOS_SSTR *pstUserInfo, ST_ZOS_SSTR *pstHostname,  
                                 ZULONG dwPort);
```

[Parameters]

**Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_ZOS\\_SSTR](#) \*pstUserName

The user name.

[ST\\_ZOS\\_SSTR](#) \*pstUserInfo

User information.

[ST\\_ZOS\\_SSTR](#) \*pstHostname

The host name.

ZULONG dwPort

The port number.

#### Output parameters:

[ST\\_SIP\\_HDR\\_REFERRED\\_BY](#) \*pstReferredBy

The referred-by header.

#### [Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.33 Sip\_HdrFillUserAgent

Fills the user agent header.

```
ZINT Sip_HdrFillUserAgent(ST\_ZOS\_DBUF *pstMemBuf,  
                          ST\_SIP\_HDR\_USER\_AGENT *pstUserAgent,  
                          ST\_ZOS\_SSTR *pstProductName, ST\_ZOS\_SSTR *pstProductVer,  
                          ST\_ZOS\_SSTR *pstComment);
```

#### [Parameters]

#### Input parameters:

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_ZOS\\_SSTR](#) \*pstProductName

The product name.

[ST\\_ZOS\\_SSTR](#) \*pstProductVer

The product version.

[ST\\_ZOS\\_SSTR](#) \*pstComment

The comment.

#### Output parameters:

[ST\\_SIP\\_HDR\\_USER\\_AGENT](#) \*pstUserAgent

The user agent header.

#### [Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.34 Sip\_HdrFillServer

Fills the server header.

```
ZINT Sip_HdrFillServer(ST\_ZOS\_DBUF *pstMemBuf,  
                      ST\_SIP\_HDR\_SERVER *pstServer, ST\_ZOS\_SSTR *pstProductName,  
                      ST\_ZOS\_SSTR *pstProductVer, ST\_ZOS\_SSTR *pstComment);
```

#### [Parameters]

##### Input parameters:

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_ZOS\\_SSTR](#) \*pstProductName

The product name.

[ST\\_ZOS\\_SSTR](#) \*pstProductVer

Product version.

[ST\\_ZOS\\_SSTR](#) \*pstComment

The comment.

##### Output parameters:

[ST SIP HDR SERVER](#) \*pstServer

The server header.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.35 Sip\_HdrFillEvt

Fills the event header.

```
ZINT Sip_HdrFillEvt(ST ZOS DBUF *pstMemBuf, ST SIP HDR EVNT *pstEvt,  
                  ST SIP EVNT PKG *pstEvtPkg);
```

[Parameters]

**Input parameters:**

[ST ZOS DBUF](#) \*pstMemBuf

The memory buffer.

[ST SIP EVNT PKG](#) \*pstEvtPkg

The event package.

**Output parameters:**

[ST SIP HDR EVNT](#) \*pstEvt

The event header.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.36 Sip\_HdrFillSubsSta

Fills the subscription state header.

```
ZINT Sip_HdrFillSubsSta(ST SIP HDR SUBS STA *pstSubsSta, ZUCHAR ucState);
```

[Parameters]

**Input parameters:**

ZUCHAR ucState

The state value.

**Output parameters:**

[ST SIP HDR SUBS STA](#) \*pstSubsSta

The subscription state header.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.37 Sip\_HdrFillRack

Fills a Rack header.

```
ZINT Sip_HdrFillRack(ST SIP HDR RACK *pstRack, ZULONG dwCseqNum,  
                    ZULONG dwRspNum, ZUCHAR ucMethod);
```

[Parameters]

**Input parameters:**

ZULONG dwCseqNum  
The sequence number.

ZULONG dwRspNum  
The response number.

ZUCHAR ucMethod  
The method name.

**Output parameters:**

[ST SIP HDR RACK](#) \*pstRack  
The Rack header.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.38 Sip\_HdrFillPrivacy

fills a Privacy header.

```
ZINT Sip_HdrFillPrivacy(ST SIP HDR PRIVACY *pstPrivacy, ZUCHAR ucPrivVal);
```

[Parameters]

**Input parameters:**

ZUCHAR ucPrivVal  
The value.

**Output parameters:**

[ST SIP HDR PRIVACY](#) \*pstPrivacy  
The Privacy header.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.39 Sip\_HdrFromToAddTag

Adds a tag to the To or From header.

```
ZINT Sip_HdrFromToAddTag(ST\_ZOS\_DBUF *pstMemBuf,  
                          ST\_SIP\_HDR\_FROM\_TO *pstFromTo, ZCHAR *pcTag, ZUSHORT wLen);
```

[Parameters]

**Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_SIP\\_HDR\\_FROM\\_TO](#) \*pstFromTo

The From or To header.

ZCHAR \*pcTag

The tag.

ZUSHORT wLen

The tag length.

**Output parameters:**

[ST\\_SIP\\_HDR\\_FROM\\_TO](#) \*pstFromTo

The From or To header.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.40 Sip\_HdrAllowAddMethod

Adds method to an Allow header.

```
ZINT Sip_HdrAllowAddMethod(ST\_ZOS\_DBUF *pstMemBuf,  
                            ST\_SIP\_HDR\_ALLOW *pstAllow, ST\_SIP\_METHOD *pstMethod);
```

[Parameters]

**Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_SIP\\_HDR\\_ALLOW](#) \*pstAllow

The Allow header.

[ST\\_SIP\\_METHOD](#) \*pstMethod

The method.

### Output parameters:

[ST\\_SIP\\_HDR\\_ALLOW](#) \*pstAllow

The Allow header.

### [Return value]

Returns ZOK on success, or ZFAILED on failure.

## 3.5.41 Sip\_HdrSuptedAddTag

Adds an option tag to a Supported header. The tag type must be one of the followings:

```

EN_SIP_OPT_TAG_100REL          /* rfc3262 */
EN_SIP_OPT_TAG_EARLY_SESS     /* rfc3959 */
EN_SIP_OPT_TAG_EVNTLST       /* rfc4662 */
EN_SIP_OPT_TAG_FROM_CHANGE   /* ietf-sip-connected-identity */
EN_SIP_OPT_TAG_HISTINFO      /* rfc4244 */
EN_SIP_OPT_TAG_JOIN          /* rfc3911 */
EN_SIP_OPT_TAG_NOREFERSUB    /* rfc4488 */
EN_SIP_OPT_TAG_PATH          /* rfc3327 */
EN_SIP_OPT_TAG_PRECONDITION  /* rfc3312 */
EN_SIP_OPT_TAG_PREF          /* rfc3480 */
EN_SIP_OPT_TAG_PRIVACY       /* rfc3323 */
EN_SIP_OPT_TAG_REPLACES      /* rfc3981 */
EN_SIP_OPT_TAG_RES_PRIORITY  /* rfc4412 */
EN_SIP_OPT_TAG_SDP_ANAT      /* rfc4092 */
EN_SIP_OPT_TAG_SEC_AGREE     /* rfc3329 */
EN_SIP_OPT_TAG_TDLG          /* rfc4538 */
EN_SIP_OPT_TAG_TMR           /* rfc4028 */
EN_SIP_OPT_TAG_MULTI_REFERER /* draft-ietf-sip-multiple-refer-01 */

```

```

ZINT Sip_HdrSuptedAddTag(ST\_ZOS\_DBUF *pstMemBuf,
                        ST\_SIP\_HDR\_SUPTED *pstSupted, ZCHAR ucType);

```

### [Parameters]

### Input parameters:

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_SIP\\_HDR\\_SUPTED](#) \*pstSupted

The Supported header.

ZCHAR ucType

The type.

#### Output parameters:

[ST\\_SIP\\_HDR\\_SUPTED](#) \*pstSupted

The Supported header.

#### [Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.42 Sip\_HdrSuptedAddTagX

Adds option tag to a Supported header. When the tag type is EN\_SIP\_OPT\_TAG\_OTHER, use this interface to add the tag.

```
ZINT Sip_HdrSuptedAddTagX(ST\_ZOS\_DBUF *pstMemBuf,
                          ST\_SIP\_HDR\_SUPTED *pstSupted, ZCHAR *pcTag, ZUSHORT wLen);
```

#### [Parameters]

##### Input parameters:

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_SIP\\_HDR\\_SUPTED](#) \*pstSupted

The Supported header.

ZCHAR \*pcTag

The tag name.

ZUSHORT wLen

Length of the tag.

##### Output parameters:

[ST\\_SIP\\_HDR\\_SUPTED](#) \*pstSupted

The Supported header.

#### [Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.43 Sip\_HdrRequireAddTag

Adds an option tag to a Require header. The tag type must be one of the followings:

```

EN_SIP_OPT_TAG_100REL          /* rfc3262 */
  EN_SIP_OPT_TAG_EARLY_SESS    /* rfc3959 */
  EN_SIP_OPT_TAG_EVNTLST      /* rfc4662 */
  EN_SIP_OPT_TAG_FROM_CHANGE  /* ietf-sip-connected-identity */
  EN_SIP_OPT_TAG_HISTINFO     /* rfc4244 */
  EN_SIP_OPT_TAG_JOIN         /* rfc3911 */
  EN_SIP_OPT_TAG_NOREFERSUB   /* rfc4488 */
  EN_SIP_OPT_TAG_PATH         /* rfc3327 */
  EN_SIP_OPT_TAG_PRECONDITION /* rfc3312 */
  EN_SIP_OPT_TAG_PREF         /* rfc3480 */
  EN_SIP_OPT_TAG_PRIVACY      /* rfc3323 */
  EN_SIP_OPT_TAG_REPLACES     /* rfc3981 */
  EN_SIP_OPT_TAG_RES_PRIORITY /* rfc4412 */
  EN_SIP_OPT_TAG_SDP_ANAT     /* rfc4092 */
  EN_SIP_OPT_TAG_SEC_AGREE    /* rfc3329 */
  EN_SIP_OPT_TAG_TDLG         /* rfc4538 */
  EN_SIP_OPT_TAG_TMR          /* rfc4028 */
  EN_SIP_OPT_TAG_MULTI_REFERER /* draft-ietf-sip-multiple-refer-01 */

```

```

ZINT Sip_HdrRequireAddTag(ST ZOS DBUF *pstMemBuf,
                          ST SIP HDR REQUIRE *pstRequire, ZCHAR ucType);

```

#### [Parameters]

##### Input parameters:

ST ZOS DBUF \*pstMemBuf

The memory buffer.

ST SIP HDR REQUIRE \*pstRequire

The Require header.

ZCHAR ucType

Type of the tag.

##### Output parameters:

ST SIP HDR REQUIRE \*pstRequire

The Require header.

#### [Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.44 Sip\_HdrRequireAddTagX

Adds an option tag to a Require header. When the tag type is EN\_SIP\_OPT\_TAG\_OTHER, use this interface to add the tag.

```
ZINT Sip_HdrRequireAddTagX(ST_ZOS_DBUF *pstMemBuf,
                           ST_SIP_HDR_REQUIRE *pstRequire, ZCHAR *pcTag, ZUSHORT wLen);
```

[Parameters]

#### Input parameters:

ST\_ZOS\_DBUF \*pstMemBuf

The memory buffer.

ST\_SIP\_HDR\_REQUIRE \*pstRequire

The Require header.

ZCHAR \*pcTag

The tag name.

ZUSHORT wLen

Length of the tag.

#### Output parameters:

ST\_SIP\_HDR\_REQUIRE \*pstRequire

The Require header.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.45 Sip\_HdrReferToAddHdr

Adds SIP URI to a Refer-To header.

```
ZINT Sip_HdrReferToAddHdr(ST_ZOS_DBUF *pstMemBuf,
                           ST_SIP_HDR_REFERER_TO *pstReferto, ZCHAR *pcName,
                           ZUSHORT wNameLen, ZCHAR *pcVal, ZUSHORT wValLen);
```

[Parameters]

#### Input parameters:

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_SIP\\_HDR\\_REFERER\\_TO](#) \*pstReferto

The Refer-To header.

ZCHAR \*pcName

The host name.

ZUSHORT wNameLen

Length of the name.

ZCHAR \*pcVal

The value.

ZUSHORT wValLen

The value length.

#### Output parameters:

[ST\\_SIP\\_HDR\\_REFERER\\_TO](#) \*pstReferto

The Refer-To header.

#### [Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.46 Sip\_HdrEvtAddId

Adds the event-param ID to an Event header.

```
ZINT Sip_HdrEvtAddId(ST\_ZOS\_DBUF *pstMemBuf,  
                    ST\_SIP\_HDR\_EVT *pstEvt, ST\_ZOS\_SSTR *pstId);
```

#### [Parameters]

##### Input parameters:

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_SIP\\_HDR\\_EVT](#) \*pstEvt

The Event header.

[ST\\_ZOS\\_SSTR](#) \*pstId

The event-param ID.

**Output parameters:**

[ST\\_SIP\\_HDR\\_EVNT](#) \*pstEvt  
The Event header.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

**3.5.47 Sip\_HdrSubsStaAddReasonVal**

Adds a reason value to a Subscription-State header.

```
ZINT Sip_HdrSubsStaAddReasonVal(ST\_ZOS\_DBUF *pstMemBuf,
                                ST\_SIP\_HDR\_SUBS\_STA *pstSubsSta, ZUCHAR ucType);
```

**[Parameters]****Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf  
The memory buffer.

[ST\\_SIP\\_HDR\\_SUBS\\_STA](#) \*pstSubsSta  
The Subscription-State header.

ZUCHAR ucType  
The reason type.

**Output parameters:**

[ST\\_SIP\\_HDR\\_SUBS\\_STA](#) \*pstSubsSta  
The Subscription-State header.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

**3.5.48 Sip\_HdrSubsStaAddExpires**

Adds expire values to a Subscription-State header.

```
ZINT Sip_HdrSubsStaAddExpires(ST\_ZOS\_DBUF *pstMemBuf,
                               ST\_SIP\_HDR\_SUBS\_STA *pstSubsSta, ZULONG dwExpire);
```

**[Parameters]****Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_SIP\\_HDR\\_SUBS\\_STA](#) \*pstSubsSta

The Subscription-State header.

ZULONG dwExpire

The expire values.

**Output parameters:**

[ST\\_SIP\\_HDR\\_SUBS\\_STA](#) \*pstSubsSta

The Subscription-State header.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

### 3.5.49 Sip\_HdrSubsStaAddRetryAfter

Adds a Retry-After value to a Subscription-State header.

```
ZINT Sip_HdrSubsStaAddRetryAfter(ST\_ZOS\_DBUF *pstMemBuf,  
                                ST\_SIP\_HDR\_SUBS\_STA *pstSubsSta, ZULONG dwRetryAfter);
```

**[Parameters]**

**Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_SIP\\_HDR\\_SUBS\\_STA](#) \*pstSubsSta

The Subscription-State header.

ZULONG dwRetryAfter

The Retry-After value.

**Output parameters:**

[ST\\_SIP\\_HDR\\_SUBS\\_STA](#) \*pstSubsSta

The Subscription-State header.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

### 3.5.50 Sip\_HdrReplaceAddFromTag

Adds a From tag to a Replaces header.

```
ZINT Sip_HdrReplaceAddFromTag(ST\_ZOS\_DBUF *pstMemBuf,  
                               ST\_SIP\_HDR\_REPLACES *pstReplace, ST\_ZOS\_SSTR *pstTag);
```

[Parameters]

**Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_SIP\\_HDR\\_REPLACES](#) \*pstReplace

The Replaces header.

[ST\\_ZOS\\_SSTR](#) \*pstTag

The tag to add.

**Output parameters:**

[ST\\_SIP\\_HDR\\_REPLACES](#) \*pstReplace

The Replaces header.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.51 Sip\_HdrReplaceAddToTag

Adds a To tag to a Replaces header.

```
ZINT Sip_HdrReplaceAddToTag(ST\_ZOS\_DBUF *pstMemBuf,  
                             ST\_SIP\_HDR\_REPLACES *pstReplace, ST\_ZOS\_SSTR *pstTag);
```

[Parameters]

**Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_SIP\\_HDR\\_REPLACES](#) \*pstReplace

The Replaces header.

[ST\\_ZOS\\_SSTR](#) \*pstTag

The tag to add.

**Output parameters:**

[ST SIP HDR REPLACES](#) \*pstReplace

The Replaces header.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.52 Sip\_HdrReplaceAddEarlyTag

Adds an Early tag to a Replaces header.

```
ZINT Sip_HdrReplaceAddEarlyTag(ST ZOS DBUF *pstMemBuf,  
                               ST SIP HDR REPLACES *pstReplace);
```

[Parameters]

**Input parameters:**

[ST ZOS DBUF](#) \*pstMemBuf

The memory buffer.

[ST SIP HDR REPLACES](#) \*pstReplace

The Replaces header.

**Output parameters:**

[ST SIP HDR REPLACES](#) \*pstReplace

The Replaces header.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.53 Sip\_HdrFromToGetTag

Gets the From or To tag from a From or To header.

```
ZINT Sip_HdrFromToGetTag(ST SIP HDR FROM TO *pstFromTo,  
                         ST ZOS SSTR **ppstTag);
```

[Parameters]

**Input parameters:**

[ST SIP HDR FROM TO](#) \*pstFromTo

The From or To header.

**Output parameters:**

[ST\\_ZOS\\_SSTR](#) \*\*ppstTag

The tag.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.54 Sip\_ParmFillStatusLine

Fills a status line.

```
ZINT Sip_ParmFillStatusLine(ST\_SIP\_STATUS\_LINE *pstStatusLine,
                             ZULONG dwStatusCode);
```

[Parameters]

**Input parameters:**

ZULONG dwStatusCode

The status code.

**Output parameters:**

[ST\\_SIP\\_STATUS\\_LINE](#) \*pstStatusLine

The status line.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.55 Sip\_ParmFillSipUri

Fills a SIP-URI or SIPS-URI.

```
ZINT Sip_ParmFillSipUri(ST\_ZOS\_DBUF *pstMemBuf, ST\_SIP\_SIP\_URI *pstSipUri,
                        ST\_ZOS\_SSTR *pstUserInfo, ST\_SIP\_HOST\_PORT *pstHostPort);
```

[Parameters]

**Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_ZOS\\_SSTR](#) \*pstUserInfo

The user information.

[ST\\_SIP\\_HOST\\_PORT](#) \*pstHostPort

The host port.

**Output parameters:**

[ST SIP SIP URI](#) \*pstSipUri  
The SIP-URI or SIPS-URI.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

**3.5.56 Sip\_ParmFillTelUri**

Fills a tel URI.

```
ZINT Sip_ParmFillTelUri(ST ZOS DBUF *pstMemBuf, ST SIP TEL URI *pstTelUri,  
                        ZBOOL bIsGlobal, ST ZOS SSTR *pstNumber);
```

**[Parameters]****Input parameters:**

[ST ZOS DBUF](#) \*pstMemBuf  
The memory buffer.

[ZBOOL](#) bIsGlobal

Indicates whether the tel URI is global. If it is global the number *pstNumber* should contain no '+'.

[ST ZOS SSTR](#) \*pstNumbe  
The number.

**Output parameters:**

[ST SIP TEL URI](#) \*pstTelUri  
The URI.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

**3.5.57 Sip\_ParmFillImUri**

Fills a instant message (IM) URI.

```
ZINT Sip_ParmFillImUri(ST ZOS DBUF *pstMemBuf, ST SIP IM URI *pstImUri,  
                       ST ZOS SSTR *pstDesc);
```

**[Parameters]**

**Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf  
The memory buffer.

[ST\\_ZOS\\_SSTR](#) \*pstDesc  
The description of the IM URI.

**Output parameters:**

[ST\\_SIP\\_IM\\_URI](#) \*pstImUri  
The IM URI.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

**3.5.58 Sip\_ParmFillReqUriByIp**

Fills a Request-URI by the IP address.

```
ZINT Sip_ParmFillReqUriByIp(ST\_ZOS\_DBUF *pstMemBuf,
                             ST\_SIP\_REQ\_URI *pstReqUri, ST\_ZOS\_SSTR *pstUserInfo,
                             ST\_ZOS\_INET\_ADDR *pstAddr);
```

**[Parameters]****Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf  
The memory buffer.

[ST\\_ZOS\\_SSTR](#) \*pstUserInfo  
User information.

[ST\\_ZOS\\_INET\\_ADDR](#) \*pstAddr  
The IP address.

**Output parameters:**

[ST\\_SIP\\_REQ\\_URI](#) \*pstReqUri  
The Request-URI.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

**3.5.59 Sip\_ParmFillReqUriByName**

Fills a Request-URI by the host name.

```
ZINT Sip_ParmFillReqUriByName(ST\_ZOS\_DBUF *pstMemBuf,  
                               ST\_SIP\_REQ\_URI *pstReqUri, ST\_ZOS\_SSTR *pstUsrInfo,  
                               ST\_ZOS\_SSTR *pstHostname, ZULONG dwPort);
```

[Parameters]

**Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_ZOS\\_SSTR](#) \*pstUserInfo

User information.

[ST\\_ZOS\\_SSTR](#) \*pstHostname

The host name.

ZULONG dwPort

The port number.

**Output parameters:**

[ST\\_SIP\\_REQ\\_URI](#) \*pstReqUri

The Request-URI.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.60 Sip\_ParmFillAddrSpecByIp

Fills an addr-spec by the IP address.

```
ZINT Sip_ParmFillAddrSpecByIp(ST\_ZOS\_DBUF *pstMemBuf,  
                               ST\_SIP\_ADDR\_SPEC *pstAddrSpec, ST\_ZOS\_SSTR *pstUserInfo,  
                               ST\_ZOS\_INET\_ADDR *pstAddr);
```

[Parameters]

**Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_ZOS\\_SSTR](#) \*pstUserInfo

The user information.

[ST\\_ZOS\\_INET\\_ADDR](#) \*pstAddr

The IP address.

#### Output parameters:

[ST\\_SIP\\_ADDR\\_SPEC](#) \*pstAddrSpec

The addr-spec.

#### [Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.61 Sip\_ParmFillAddrSpecByName

Fills an addr-spec by the host name.

```
ZINT Sip_ParmFillAddrSpecByName(ST\_ZOS\_DBUF *pstMemBuf,
                                ST\_SIP\_ADDR\_SPEC *pstAddrSpec, ST\_ZOS\_SSTR *pstUserInfo,
                                ST\_ZOS\_SSTR *pstHostname, ZULONG dwPort);
```

#### [Parameters]

##### Input parameters:

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_ZOS\\_SSTR](#) \*pstUserInfo

The user information.

[ST\\_ZOS\\_SSTR](#) \*pstHostname

The host name.

ZULONG dwPort

The port number.

##### Output parameters:

[ST\\_SIP\\_ADDR\\_SPEC](#) \*pstAddrSpec

The addr-spec.

#### [Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.62 Sip\_ParmFillAddrSpecBySipUri

Fills an addr-spec by the SIP URI.

```
ZINT Sip_ParmFillAddrSpecBySipUri(ST\_ZOS\_DBUF *pstMemBuf,  
                                   ST\_SIP\_ADDR\_SPEC *pstAddrSpec, ST\_SIP\_SIP\_URI *pstSipUri);
```

[Parameters]

**Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_SIP\\_SIP\\_URI](#) \*pstSipUri

The SIP URI.

**Output parameters:**

[ST\\_SIP\\_ADDR\\_SPEC](#) \*pstAddrSpec

The addr-spec.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.63 Sip\_ParmFillAddrSpecBySipsUri

Fills an addr-spec by the SIPS URI.

```
ZINT Sip_ParmFillAddrSpecBySipsUri(ST\_ZOS\_DBUF *pstMemBuf,  
                                    ST\_SIP\_ADDR\_SPEC *pstAddrSpec, ST\_SIP\_SIP\_URI *pstSipsUri);
```

[Parameters]

**Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_SIP\\_SIP\\_URI](#) \*pstSipsUri

The SIPS URI.

**Output parameters:**

[ST SIP ADDR SPEC](#) \*pstAddrSpec

The addr-spec.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.64 Sip\_ParmFillAddrSpecByTelUri

Fills a addr-spec by the tel URI.

```
ZINT Sip_ParmFillAddrSpecByTelUri(ST ZOS DBUF *pstMemBuf,  
                                   ST SIP ADDR SPEC *pstAddrSpec, ST SIP TEL URI *pstTelUri);
```

[Parameters]

**Input parameters:**

[ST ZOS DBUF](#) \*pstMemBuf

The memory buffer.

[ST SIP TEL URI](#) \*pstTelUri

The tel URI.

**Output parameters:**

[ST SIP ADDR SPEC](#) \*pstAddrSpec

The addr-spec.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.65 Sip\_ParmFillAddrSpecByImUri

Fills an addr-spec by the IM URI.

```
ZINT Sip_ParmFillAddrSpecByImUri(ST ZOS DBUF *pstMemBuf,  
                                   ST SIP ADDR SPEC *pstAddrSpec, ST SIP IM URI *pstImUri);
```

[Parameters]

**Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_SIP\\_IM\\_URI](#) \*pstImUri

The IM URI.

#### Output parameters:

[ST\\_SIP\\_ADDR\\_SPEC](#) \*pstAddrSpec

The addr-spec.

#### [Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.66 Sip\_ParmFillContactParm

Fills the contact-param.

```
ZINT Sip_ParmFillContactParm(ST\_ZOS\_DBUF *pstMemBuf,  
                             ST\_SIP\_CONTACT\_PARM *pstContactParm,  
                             ST\_SIP\_ADDR\_SPEC *pstAddrSpec,  
                             ST\_ZOS\_INET\_ADDR *pstInetAddr);
```

#### [Parameters]

#### Input parameters:

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_SIP\\_ADDR\\_SPEC](#) \*pstAddrSpec

The addr-spec.

[ST\\_ZOS\\_INET\\_ADDR](#) \*pstInetAddr

The IP address.

#### Output parameters:

[ST\\_SIP\\_CONTACT\\_PARM](#) \*pstContactParm

The contact-param.

#### [Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.67 Sip\_ParmFillHostPort

Fills the host port.

```
ZINT Sip_ParmFillHostPort(ST SIP HOST PORT *pstHostPort,  
                          ST ZOS INET ADDR *pstAddr);
```

[Parameters]

**Input parameters:**

[ST ZOS INET ADDR](#) \*pstAddr  
The IP address.

**Output parameters:**

[ST SIP HOST PORT](#) \*pstHostPort  
The host port.

[Return value]

Returns ZOK.

### 3.5.68 Sip\_ParmFillViaSentProtocol

Fills the sent-protocol in a via parameter.

```
ZINT Sip_ParmFillViaSentProtocol(ST ZOS DBUF *pstMemBuf,  
                                 ST SIP VIA PARM *pstViaParm, ZUCHAR ucTptType);
```

[Parameters]

**Input parameters:**

[ST ZOS DBUF](#) \*pstMemBuf  
The memory buffer.

ZUCHAR ucTptType  
The transport type.

**Output parameters:**

[ST SIP VIA PARM](#) \*pstViaParm  
The via parameter.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.69 Sip\_ParmFillViaSentBy

Fills the sent-by in a via parameter.

```
ZINT Sip_ParmFillViaSentBy( ST SIP VIA PARM *pstViaParm,  
ST ZOS INET ADDR *pstAddr);
```

[Parameters]

**Input parameters:**

[ST ZOS INET ADDR](#) \*pstAddr

The IP address.

**Output parameters:**

[ST SIP VIA PARM](#) \*pstViaParm

The via parameter.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.70 Sip\_ParmFillViaBranch

Sets the via-branch in a via parameter.

```
ZINT Sip_ParmFillViaBranch(ST ZOS DBUF *pstMemBuf,  
ST SIP VIA PARM *pstViaParm, ST ZOS SSTR *pstBranch);
```

[Parameters]

**Input parameters:**

[ST ZOS DBUF](#) \*pstMemBuf

The memory buffer.

[ST ZOS SSTR](#) \*pstBranch

The string which the via-branch to be set.

**Output parameters:**

[ST SIP VIA PARM](#) \*pstViaParm

The via parameter.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.71 Sip\_ParmFillViaRecv

Sets the via-received in a via parameter.

```
ZINT Sip_ParmFillViaRecv(ST\_ZOS\_DBUF *pstMemBuf,  
                        ST\_SIP\_VIA\_PARM *pstViaParm, ST\_ZOS\_INET\_ADDR *pstAddr);
```

#### [Parameters]

##### Input parameters:

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_ZOS\\_INET\\_ADDR](#) \*pstAddr

The IP address.

##### Output parameters:

[ST\\_SIP\\_VIA\\_PARM](#) \*pstViaParm

The via parameter.

#### [Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.72 Sip\_ParmFillViaRport

Sets the response-port in a via parameter.

```
ZINT Sip_ParmFillViaRport(ST\_ZOS\_DBUF *pstMemBuf,  
                        ST\_SIP\_VIA\_PARM *pstViaParm, ZULONG dwRport);
```

#### [Parameters]

##### Input parameters:

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

ZULONG dwRport

The port number.

##### Output parameters:

[ST\\_SIP\\_VIA\\_PARM](#) \*pstViaParm

The via parameter.

#### [Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.73 Sip\_ParmFillEvtType

Fills a event type.

```
ZINT Sip_ParmFillEvtType(ST_ZOS_DBUF *pstMemBuf,
                        ST_SIP_EVNT_TYPE *pstEvtType, ST_SIP_EVNT_PKG *pstEvtPkg);
```

## [Parameters]

**Input parameters:**

ST\_ZOS\_DBUF \*pstMemBuf

The memory buffer.

ST\_SIP\_EVNT\_PKG \*pstEvtPkg

The event package.

**Output parameters:**

ST\_SIP\_EVNT\_TYPE \*pstEvtType

The event type.

## [Return value]

Returns ZOK on success, or ZFAILED on failure.

**3.5.74 Sip\_ParmFillEvtPkg**

Sets the type of an event package when the type is among the followings:

```
EN_SIP_EVNT_PKG_PRESENCE      /* rfc3856 */
  EN_SIP_EVNT_PKG_REFER      /* rfc3515 */
  EN_SIP_EVNT_PKG_POC_SET    /* rfc4354 */
  EN_SIP_EVNT_PKG_CONF      /* rfc4575 */
  EN_SIP_EVNT_PKG_DLG      /* rfc4235 */
  EN_SIP_EVNT_PKG_KPML      /* rfc4730 */
  EN_SIP_EVNT_PKG_MSG_SUMMAR, /* rfc3842 */
  EN_SIP_EVNT_PKG_REG      /* rfc3680 */
  EN_SIP_EVNT_PKG_UA_PROF    /* draft-ietf-sip-xcap-config */
```

```
ZINT Sip_ParmFillEvtPkg(ST_SIP_EVNT_PKG *pstEvtPkg, ZUCHAR ucType);
```

## [Parameters]

**Input parameters:**

ZUCHAR ucType

The package type.

**Output parameters:**

ST\_SIP\_EVNT\_PKG \*pstEvtPkg

The event package.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

**3.5.75 Sip\_ParmFillEvtPkgX**

Sets the type of an event package when the type is not among that mentioned above, meaning when the type is EN\_SIP\_EVT\_PKG\_OTHER.

```
ZINT Sip_ParmFillEvtPkgX(ST\_ZOS\_DBUF *pstMemBuf,
                        ST\_SIP\_EVT\_PKG *pstEvtPkg, ZCHAR *pcOther, ZUSHORT wLen);
```

**[Parameters]****Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

ZCHAR \*pcOther

The type name.

ZUSHORT wLen

Length of the type name.

**Output parameters:**

[ST\\_SIP\\_EVT\\_PKG](#) \*pstEvtPkg

The event package.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

**3.5.76 Sip\_ParmFillEvtTemp**

Sets an event template.

```
ZINT Sip_ParmFillEvtTemp(ST\_ZOS\_DBUF *pstMemBuf,
                        ST\_SIP\_EVT\_TYPE *pstEvtPkg, ZUCHAR ucType);
```

**[Parameters]****Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

ZUCHAR ucType

The template type.

**Output parameters:**

[ST\\_SIP\\_EVT\\_TYPE](#) \*pstEvtPkg

The event type.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

### 3.5.77 Sip\_ParmFillDispName

Sets a display name.

```
ZINT Sip_ParmFillDispName(ST\_ZOS\_DBUF *pstMemBuf,  
                           ST\_SIP\_DISP\_NAME *pstDispName, ST\_ZOS\_SSTR *pstName);
```

**[Parameters]**

**Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_ZOS\\_SSTR](#) \*pstName

The display name.

**Output parameters:**

[ST\\_SIP\\_DISP\\_NAME](#) \*pstDispName

The display name.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

### 3.5.78 Sip\_ParmFillCredents

Fills credentials based on the challenge.

```
ZINT Sip_ParmFillCredents(ST\_ZOS\_DBUF *pstMemBuf,
                          ST\_SIP\_CREDENTIALS *pstCredents, ZUCHAR ucMethod,
                          ST\_SIP\_CHALLENGE *pstChallenge, ST\_ZOS\_SSTR *pstUser,
                          ST\_ZOS\_SSTR *pstPass, ST\_SIP\_REQ\_URI *pstDigestUri);
```

[Parameters]

**Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

ZUCHAR ucMethod

The method.

[ST\\_SIP\\_CHALLENGE](#) \*pstChallenge

The challenge.

[ST\\_ZOS\\_SSTR](#) \*pstUser

The user.

[ST\\_ZOS\\_SSTR](#) \*pstPass

The password.

[ST\\_SIP\\_REQ\\_URI](#) \*pstDigestUri

The digest URI.

**Output parameters:**

[ST\\_SIP\\_CREDENTIALS](#) \*pstCredents

The credentials.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.79 Sip\_ParmFillDRspUserName

Sets the user name in a digest response.

```
ZINT Sip_ParmFillDRspUserName(ST\_ZOS\_DBUF *pstMemBuf,
                              ST\_SIP\_DIGEST\_RSP *pstDigestRsp, ST\_ZOS\_SSTR *pstUser);
```

[Parameters]

**Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_ZOS\\_SSTR](#) \*pstUser

The user name.

**Output parameters:**

[ST\\_SIP\\_DIGEST\\_RSP](#) \*pstDigestRsp

The digest response.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

### 3.5.80 Sip\_ParmFillDRspRealm

Sets the realm in a digest response.

```
ZINT Sip_ParmFillDRspRealm(ST\_ZOS\_DBUF *pstMemBuf,  
                           ST\_SIP\_DIGEST\_RSP *pstDigestRsp, ST\_ZOS\_SSTR *pstRealm);
```

**[Parameters]**

**Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_ZOS\\_SSTR](#) \*pstRealm

The realm to copy.

**Output parameters:**

[ST\\_SIP\\_DIGEST\\_RSP](#) \*pstDigestRsp

The digest response.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

### 3.5.81 Sip\_ParmFillDRspNonce

Sets the Nonce in a digest response.

```
ZINT Sip_ParmFillDRspNonce(ST\_ZOS\_DBUF *pstMemBuf,  
                           ST\_SIP\_DIGEST\_RSP *pstDigestRsp, ST\_ZOS\_SSTR *pstNonce);
```

**[Parameters]**

**Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf  
The memory buffer.

[ST\\_ZOS\\_SSTR](#) \*pstNonce  
The Nonce.

**Output parameters:**

[ST\\_SIP\\_DIGEST\\_RSP](#) \*pstDigestRsp  
The digest response.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

**3.5.82 Sip\_ParmFillDRspUri**

Sets the Request-URI in a digest response.

```
ZINT Sip_ParmFillDRspUri(ST\_ZOS\_DBUF *pstMemBuf,
                        ST\_SIP\_DIGEST\_RSP *pstDigestRsp, ST\_SIP\_REQ\_URI *pstDigestUri);
```

**[Parameters]****Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf  
The memory buffer.

[ST\\_SIP\\_REQ\\_URI](#) \*pstDigestUri  
The Request-URI.

**Output parameters:**

[ST\\_SIP\\_DIGEST\\_RSP](#) \*pstDigestRsp  
The digest response.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

**3.5.83 Sip\_ParmFillDRspRsp**

Sets the response MD5 in a digest response.

```
ZINT Sip_ParmFillDRspRsp(ST\_ZOS\_DBUF *pstMemBuf,
                        ST\_SIP\_DIGEST\_RSP *pstDigestRsp, ZCHAR *pcRspStr);
```

## [Parameters]

**Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

ZCHAR \*pcRspStr

The response MD5.

**Output parameters:**

[ST\\_SIP\\_DIGEST\\_RSP](#) \*pstDigestRsp

The digest response.

## [Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.84 Sip\_ParmFillDRspOpaque

Sets the Opaque in a digest response.

```
ZINT Sip_ParmFillDRspOpaque(ST\_ZOS\_DBUF *pstMemBuf,  
                             ST\_SIP\_DIGEST\_RSP *pstDigestRsp, ST\_ZOS\_SSTR *pstOpaque);
```

## [Parameters]

**Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_ZOS\\_SSTR](#) \*pstOpaque

The Opqaue.

**Output parameters:**

[ST\\_SIP\\_DIGEST\\_RSP](#) \*pstDigestRsp

The digest response.

## [Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.85 Sip\_ParmFillDRspAlgo

Sets the algorithm in a digest response.

```
ZINT Sip_ParmFillDRspAlgo(ST\_ZOS\_DBUF *pstMemBuf,  
                          ST\_SIP\_DIGEST\_RSP *pstDigestRsp, ST\_SIP\_ALGO *pstAlgo);
```

#### [Parameters]

##### Input parameters:

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

[ST\\_SIP\\_ALGO](#) \*pstAlgo

The algorithm.

##### Output parameters:

[ST\\_SIP\\_DIGEST\\_RSP](#) \*pstDigestRsp

The digest response.

#### [Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.86 Sip\_ParmFillContactIsFocus

Adds a contact parameter of isfocus to a contact parameter list.

```
ZINT Sip_ParmFillContactIsFocus(ST\_ZOS\_DBUF *pstMemBuf,  
                                ST\_SIP\_CONTACT\_PARM *pstContactParm);
```

#### [Parameters]

##### Input parameters:

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

##### Output parameters:

[ST\\_SIP\\_CONTACT\\_PARM](#) \*pstContactParm

The contact parameter list.

#### [Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.87 Sip\_ParmFillContactPocTb

Adds a +g.poc.talkburst parameter to a contact parameter list.

```
ZINT Sip_ParmFillContactPocTb(ST_ZOS_DBUF *pstMemBuf,  
                               ST_SIP_CONTACT_PARM *pstContactParm);
```

[Parameters]

**Input parameters:**

ST\_ZOS\_DBUF \*pstMemBuf

The memory buffer.

**Output parameters:**

ST\_SIP\_CONTACT\_PARM \*pstContactParm

The contact parameter list.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.88 Sip\_ParmFillContact3gppCv

Adds the +g.3gpp.cs-voice to a contact parameter list.

```
ZINT Sip_ParmFillContact3gppCv(ST_ZOS_DBUF *pstMemBuf,  
                               ST_SIP_CONTACT_PARM *pstContactParm);
```

[Parameters]

**Input parameters:**

ST\_ZOS\_DBUF \*pstMemBuf

The memory buffer.

**Output parameters:**

ST\_SIP\_CONTACT\_PARM \*pstContactParm

The contact parameter list.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.89 Sip\_ParmFillAcValPocTb

Adds a +g.poc.talkburst parameter to an ac-value parameter list.

```
ZINT Sip_ParmFillAcValPocTb(ST_ZOS_DBUF *pstMemBuf,  
                             ST_SIP_AC_VAL *pstAcVal);
```

[Parameters]

**Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

#### Output parameters:

[ST\\_SIP\\_AC\\_VAL](#) \*pstAcVal

The ac-value parameter list.

#### [Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.90 Sip\_ParmFillAcVal3gppCv

Adds a +g.3gpp.cs-voice into an ac-value parameter list.

```
ZINT Sip_ParmFillAcVal3gppCv(ST\_ZOS\_DBUF *pstMemBuf,
                             ST\_SIP\_AC\_VAL *pstAcVal);
```

#### [Parameters]

#### Input parameters:

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

#### Output parameters:

[ST\\_SIP\\_AC\\_VAL](#) \*pstAcVal

The ac-value parameter list.

#### [Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.91 Sip\_ParmFillAcValRequire

Adds a Require parameter to an ac-value parameter list.

```
ZINT Sip_ParmFillAcValRequire(ST\_ZOS\_DBUF *pstMemBuf,
                              ST\_SIP\_AC\_VAL *pstAcVal);
```

#### [Parameters]

#### Input parameters:

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

#### Output parameters:

[ST SIP AC VAL](#) \*pstAcVal

The ac-value parameter list.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.92 Sip\_ParmFillAcValExplicit

Adds an Explicit parameter to an ac-value parameter list.

```
ZINT Sip_ParmFillAcValExplicit(ST ZOS DBUF *pstMemBuf,  
                               ST SIP AC VAL *pstAcVal);
```

[Parameters]

**Input parameters:**

[ST ZOS DBUF](#) \*pstMemBuf

The memory buffer.

**Output parameters:**

[ST SIP AC VAL](#) \*pstAcVal

The ac-value parameter list.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.93 Sip\_ParmFillMediaType

Fills a media type.

```
ZINT Sip_ParmFillMediaType(ST SIP MEDIA TYPE *pstMediaType,  
                           ZUCHAR ucMtype, ZUCHAR ucMSubtype);
```

[Parameters]

**Input parameters:**

ZUCHAR ucMtype

The m-type type.

ZUCHAR ucMSubtype

The m-subtype type.

**Output parameters:**

[ST SIP MEDIA TYPE](#) \*pstMediaType

The media type.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

**3.5.94 Sip\_ParmOptTagLstAddTag**

Adds a tag to an option-tag list.

```
ZINT Sip_ParmOptTagLstAddTag(ST_ZOS_DBUF *pstMemBuf,
                             ST_SIP_OPT_TAG_LST *pstOptTagLst, ZUCHAR ucType);
```

**[Parameters]****Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

ZUCHAR ucType

Type of the tag to add.

**Output parameters:**

ST\_SIP\_OPT\_TAG\_LST \*pstOptTagLst

The option-tag list.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

**3.5.95 Sip\_ParmOptTagLstGetTag**

Gets the first tag of a particular type from an option-tag list.

```
ZINT Sip_ParmOptTagLstGetTag(ST_SIP_OPT_TAG_LST *pstOptTagLst,
                             ZUCHAR ucType, ST\_SIP\_OPT\_TAG **ppstTag);
```

**Parameters]****Input parameters:**

ST\_SIP\_OPT\_TAG\_LST \*pstOptTagLst

The option-tag list

ZUCHAR ucType

Type of the tag to get.

**Output parameters:**

[ST\\_SIP\\_OPT\\_TAG](#) \*\*ppstTag

The tag on success.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

### 3.5.96 Sip\_ParmFromToLstAddTag

Adds a tag to a From or To parameter list.

```
ZINT Sip_ParmFromToLstAddTag(ST_ZOS_DBUF *pstMemBuf,  
                             ST_SIP_FROM_TO_PARM_LST *pstParmLst,  
                             ZCHAR *pcTag, ZUSHORT wLen);
```

**[Parameters]****Input parameters:**

[ST\\_ZOS\\_DBUF](#) \*pstMemBuf

The memory buffer.

ZCHAR \*pcTag

The tag to add.

ZUSHORT wLen

Length of the tag.

**Output parameters:**

ST\_SIP\_FROM\_TO\_PARM\_LST \*pstParmLst

The From or To parameter list.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

### 3.5.97 Sip\_ParmFromToLstGetTag

Gets a tag from a From or To parameter list.

```
ZINT Sip_ParmFromToLstGetTag(ST_SIP_FROM_TO_PARM_LST *pstParmLst,  
                             ST\_ZOS\_SSTR **ppstTag);
```

**[Parameters]****Input parameters:**

ST\_SIP\_FROM\_TO\_PARM\_LST \*pstParmLst

The From or To parameter list.

**Output parameters:**

[ST\\_ZOS\\_SSTR](#) \*\*ppstTag

The tag on success.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.98 Sip\_ParmEvtLstGetId

Gets the first ID parameter from an event-parameter list.

```
ZINT Sip_ParmEvtLstGetId(ST_SIP_EVNT_PARM_LST *pstParmLst,  
                          ST\_ZOS\_SSTR **ppstId);
```

[Parameters]

**Input parameters:**

ST\_SIP\_EVNT\_PARM\_LST \*pstParmLst

The event-parameter list.

**Output parameters:**

[ST\\_ZOS\\_SSTR](#) \*\*ppstId

The ID on success.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.99 Sip\_ParmContactLstGetExpire

Gets the first Expire parameter from a contact-parameter list.

```
ZINT Sip_ParmContactLstGetExpire(ST_SIP_CONTACT_PARAMS_LST *pstParmLst,  
                                   ZULONG *pdwExpire);
```

[Parameters]

**Input parameters:**

ST\_SIP\_CONTACT\_PARAMS\_LST \*pstParmLst

The contact-parameter list.

**Output parameters:**

ZULONG \*pdwExpire

The Expire parameter on success.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.100 Sip\_ParmMethodLstGetMethod

Gets the first method of a particular from a method list.

```
ZINT Sip_ParmMethodLstGetMethod(ST_SIP_METHOD_LST *pstMethodLst,  
                                ZUCHAR ucType, ST\_SIP\_METHOD **ppstMethod);
```

[Parameters]

**Input parameters:**

ST\_SIP\_METHOD\_LST \*pstMethodLst  
The method list.

ZUCHAR ucType  
Type of the method to get.

**Output parameters:**

[ST\\_SIP\\_METHOD](#) \*\*ppstMethod  
The method on success.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.5.101 Sip\_ParmMediaLstAddAttr

Adds an attribute to a media parameter list.

```
ZINT Sip_ParmMediaLstAddAttr(ST\_ZOS\_DBUF *pstMemBuf,  
                             ST_SIP_MPARAM_LST *pstParmLst, ZCHAR *pcAttr, ZUSHORT wAttrLen,  
                             ZCHAR *pcVal, ZUSHORT wValLen);
```

[Parameters]

**Input parameters:**

`ST_ZOS_DBUF *pstMemBuf`  
The memory buffer.

`ST_SIP_MPARAM_LST *pstParmLst`  
The media parameter list.

`ZCHAR *pcAttr`  
The attribute to add.

`ZUSHORT wAttrLen`  
Length of the attribute.

`ZCHAR *pcVal`  
The media value string.

`ZUSHORT wValLen`  
Length of the media value string.

**Output parameters:**

None.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

## 3.6 Utility Interfaces

### 3.6.1 Sip\_LogOpen

Opens log files.

```
ZINT Sip_LogOpen(ZULONG dwLogMod, ZULONG dwLogLevel)
```

**[Parameters]**

Input parameters:

`ZULONG dwLogMod`  
Indicates whether to open all the log files or just a few of them.

`ZULONG dwLogLevel`  
The log level.

**Output parameters:**

None.

**[Return value]**

Returns ZOK on success, or ZFAILED on failure.

### 3.6.2 Sip\_LogClose

Closes log files.

```
ZINT Sip_LogClose(ZULONG dwLogMod, ZULONG dwLogLevel)
```

#### [Parameters]

Input parameters:

ZULONG dwLogMod

Indicates which log files to close.

ZULONG dwLogLevel

The log level.

#### Output parameters:

None.

#### [Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.6.3 Sip\_SessEvtFree

Frees a session event.

```
ZINT Sip_SessEvtFree(ST SIP SESS EVNT *pstEvt);
```

#### [Parameters]

Input parameters:

[ST SIP SESS EVNT](#) \*pstEvt

The session event.

#### Output parameters:

None.

#### [Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.6.4 Sip\_ReasonFromCode

Gets the status reason phrase from a status code.

```
ZINT Sip_ReasonFromCode(ZULONG dwCode, ST\_ZOS\_SSTR *pstReason);
```

#### [Parameters]

##### Input parameters:

ZULONG dwCode  
The status code.

##### Output parameters:

[ST\\_ZOS\\_SSTR](#) \*pstReason  
The status reason.

#### [Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.6.5 Sip\_GetMethodDesc

Gets a method description.

```
ZCHAR * Sip_GetMethodDesc(ZUCHAR ucMethod);
```

#### [Parameters]

##### Input parameters:

ZUCHAR ucMethod  
The method.

##### Output parameters:

None.

#### [Return value]

Returns the description.

### 3.6.6 Sip\_GetSessEvntDesc

Gets the description of a session event.

```
ZCHAR * Sip_GetSessEvntDesc(ZUCHAR ucEvntType);
```

#### [Parameters]

##### Input parameters:

ZUCHAR ucEvntType  
The session event.

##### Output parameters:

None.

[Return value]

Returns the description.

### 3.6.7 Sip\_GetStatCodeDesc

Gets the description of a status code.

```
ZCHAR * Sip_GetStatCodeDesc (ZULONG dwStatCode);
```

[Parameters]

Input parameters:

```
ZULONG dwStatCode  
The status code.
```

**Output parameters:**

None.

[Return value]

Returns the description.

## 3.7 Task Interfaces

These interfaces are included in sip\_task.h.

### 3.7.1 Sip\_Start

Starts the SIP task.

```
ZINT Sip_Start();
```

[Parameters]

Input parameters:

None.

**Output parameters:**

None.

[Return value]

Returns ZOK on success, or ZFAILED on failure.

### 3.7.2 Sip\_Stop

Stops the SIP task.

```
ZVOID Sip_Stop();
```

[Parameters]

Input parameters:

None.

**Output parameters:**

None.

[Return value]

None.

### 3.7.3 Sip\_Restart

Restarts the SIP task.

```
ZINT Sip_Restart();
```

[Parameters]

Input parameters:

None.

**Output parameters:**

None.

[Return value]

Returns ZOK.

## 3.8 Version Interfaces

These interfaces are included in

### 3.8.1 Sip\_GetVersion

Gets the version of SIP.

```
ZCHAR * Sip_GetVersion();
```

[Parameters]

Input parameters:

None.

**Output parameters:**

None.

**[Return value]**

Returns the version string of SIP.

## 4. Interface Procedures

This chapter presents the interface procedures that are unrelated to a specific SIP session.

### 4.1 Normal SIP Procedure

Figure 4-1 shows a normal SIP procedure between two user agents.

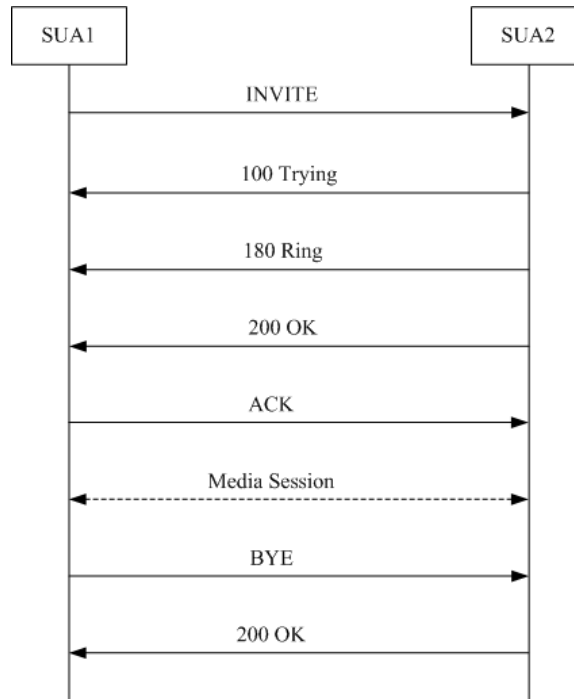


Figure 4-1 A Normal SIP Procedure between Two User Agents

### 4.2 Successful Outgoing Call Setup

Figure 4-2 shows a normal SIP call setup.

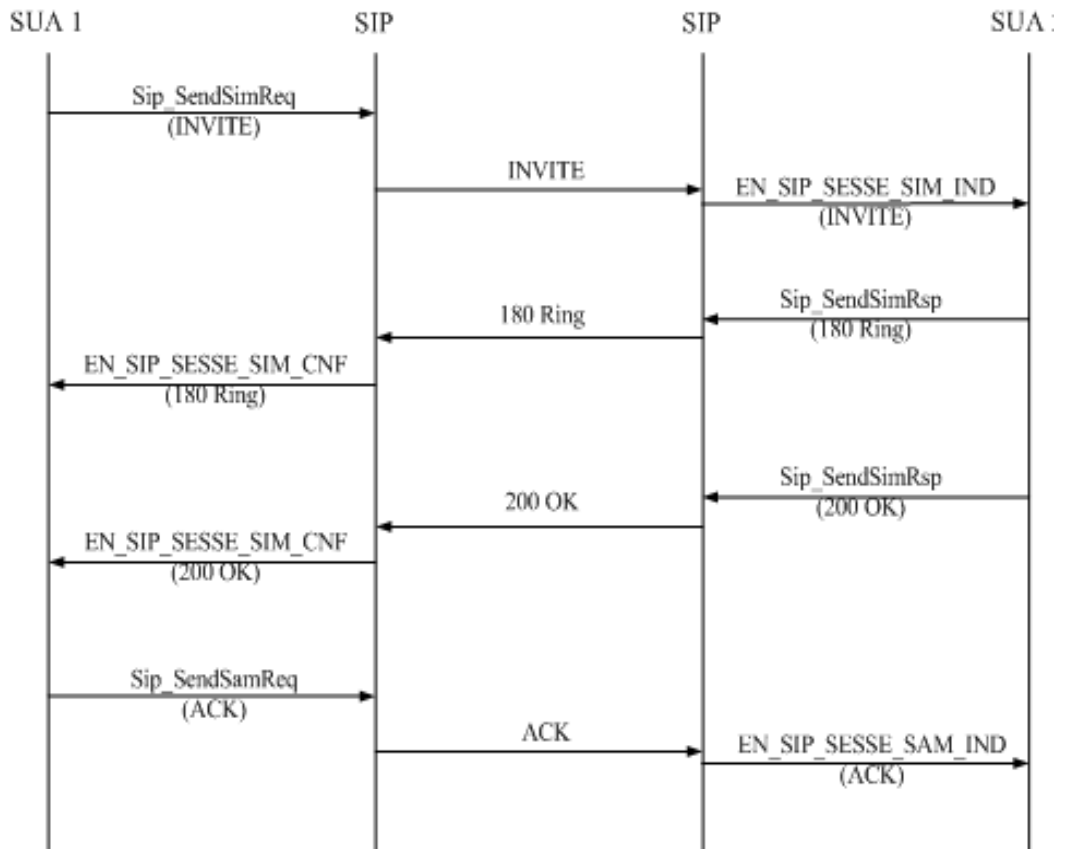


Figure 4-2 Successful Outgoing SIP Call Setup

### 4.3 Successful Incoming SIP Call

Figure 4-3 shows the procedure of a successful incoming SIP call setup.

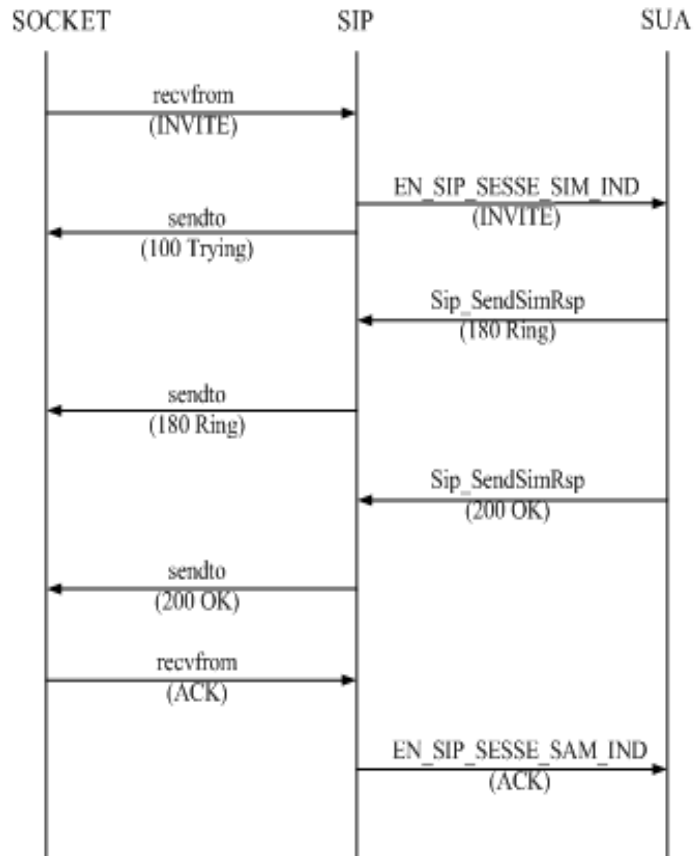


Figure 4-3 Successful Incoming SIP Call Setup

#### 4.4 Session Status Request

Figure 4-4 shows the procedure of sending a session status request.

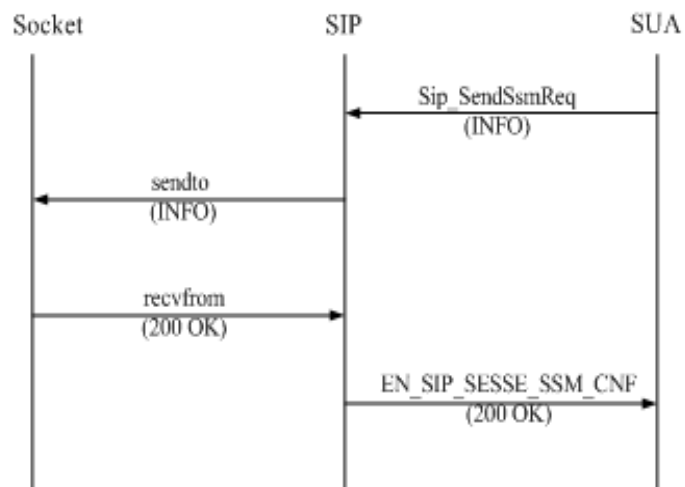


Figure 4-4 Session Status Request

### 4.5 Session Status Response

Figure 4-5 shows the procedure of sending a response about session status.

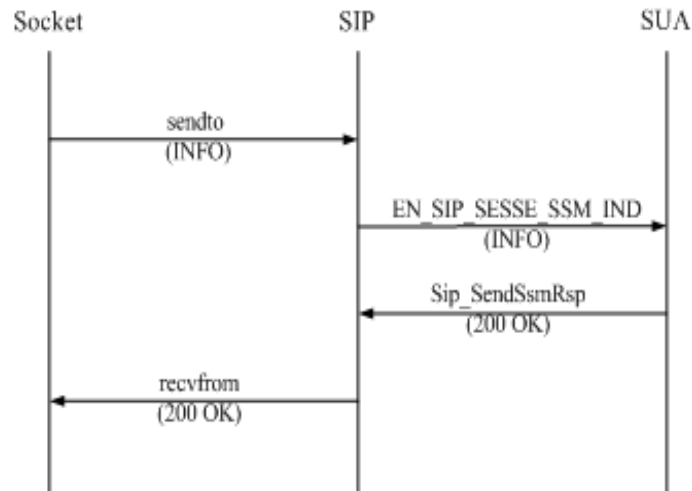


Figure 4-5 Session Status Response

### 4.6 Re-INVITE Request

Figure 4-6 shows the procedure of sending a re-INVITE request.

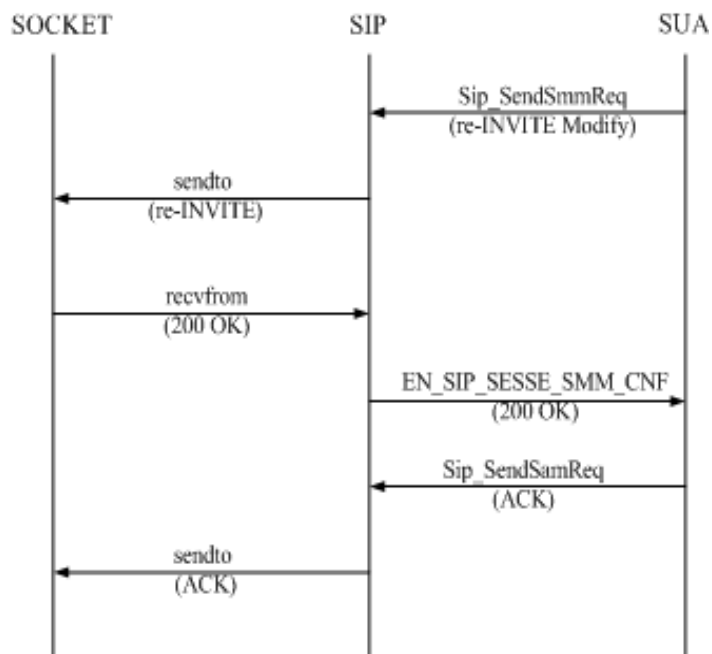


Figure 4-6 A re-INVITE Request

### 4.7 Response to re-INVITE Request

Figure 4-7 shows the procedure of sending a response to a re-INVITE request.

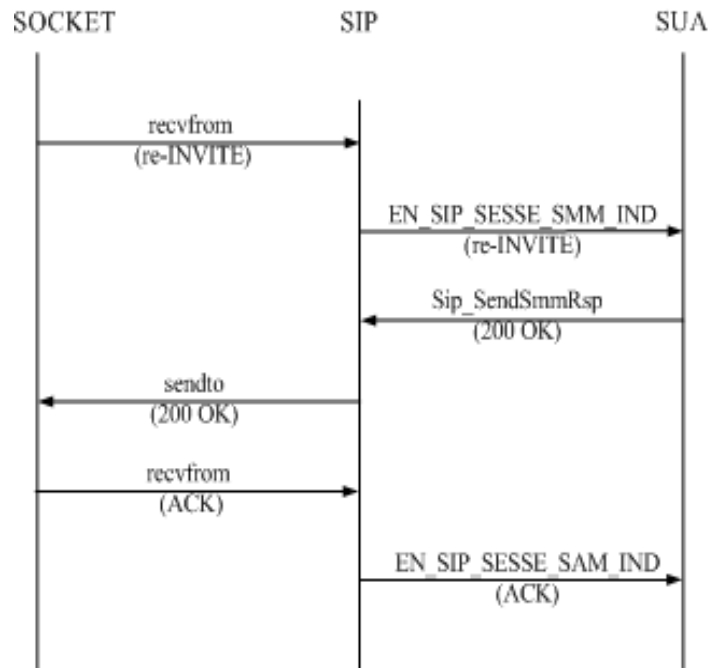


Figure 4-7 The Reponse to a re-INVITE Request

### 4.8 BYE Request

Figure 4-8 shows the procedure of sending a BYE request.

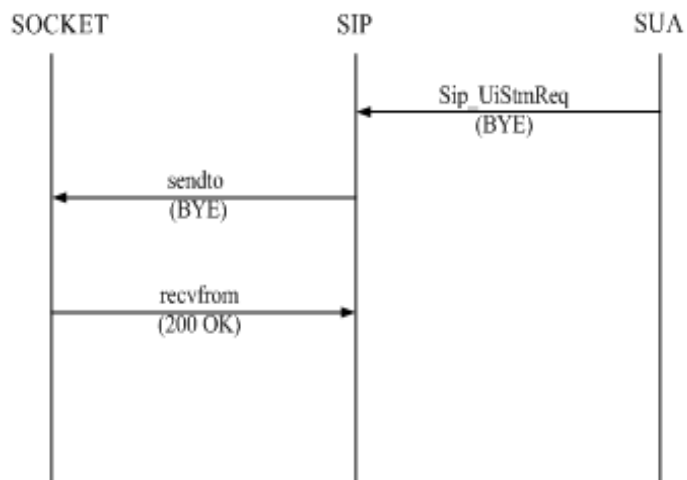


Figure 4-8 A BYE Request

### 4.9 Response to BYE Request

Figure 4-9 shows the procedure of sending a response to a BYE request.

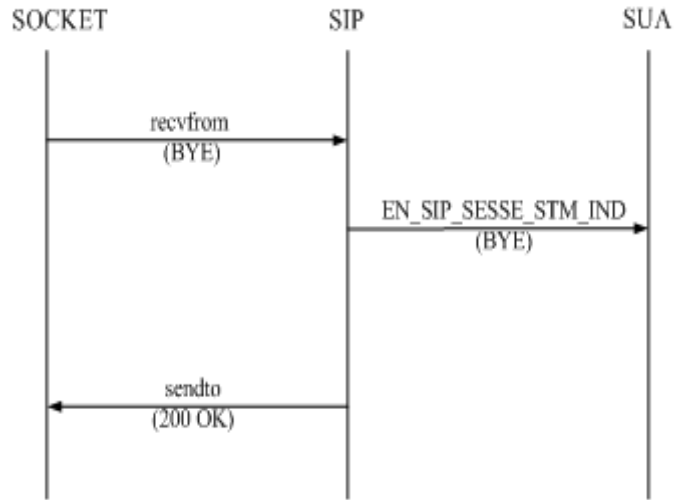


Figure 4-9 The Reponse to a BYE Request

## 5. Interface Examples

Here are some examples on how to use the user interfaces.

### 5.1 Test SIP structures

```

/* test sip customer setting */
typedef struct tagTSIP_CTM_SET
{
    ZCHAR *pcName;           /* local name in sip message */
    ZCHAR *pcIp;            /* local ip in sip message */
    ZUSHORT wPort;          /* local port in sip message */
} ST_TSIP_CTM_SET;

/* test SIP call info */
typedef struct tagTSIP_CALL_INFO
{
    ZULONG dwUserId;        /* user ID */
    ZULONG dwSessId;        /* session ID */
    ZULONG dwDlgId;         /* dialog ID */
    ZULONG dwTransId;       /* transaction ID */
    ST_SIP_MSG *pstLastMsg; /* last message */
    ZULONG dwTesterCseq;    /* cseq value of tester */
    ST_ZOS_INET_ADDR stRmtAddr; /* remote ip address */
    ZUCHAR ucMethodType;    /* method type EN_SIP_METHOD */
    ZUCHAR ucSubsSta;       /* subscription state EN_SIP_SUBSTA_VAL */
    ZUCHAR ucIsTester;      /* is tester */
    ZUCHAR aucSpare[1];     /* for 32 bit alignment */
    ZULONG dwRspCode;       /* response code */
    ZGABID zGabId;          /* memory resource garbage bin */
    ST_ZOS_SSTR stEvent;    /* event for subscription */
    ZBOOL bUseCtmName;      /* flag for use name set by user */
    ST_TSIP_CTM_SET stLocalSet; /* customer setting for local */
    ST_TSIP_CTM_SET *pstRemoteSet; /* customer setting for remote */
} ST_TSIP_CALL_INFO;

```

### 5.2 Test SIP macros

```
/* test SIP create dbuf */
#define TSIP_DBUF_CREATE(_uctype, _dftblksize) \
    Zos_DbufCreate(ZNULL, _uctype, _dftblksize)

/* test SIP alloc memory of specific size from dbuf */
#define TSIP_DBUF_ALLOC(_dbuf, _size) \
    Zos_DbufAlloc(_dbuf, _size)

/* test SIP alloc memory(zeroed) of specific size from dbuf */
#define TSIP_DBUF_ALLOC_CLRD(_dbuf, _size) \
    Zos_DbufAllocClrd(_dbuf, _size)

/* test SIP delete dbuf */
#define TSIP_DBUF_DELETE(_dbuf) do { \
    if (_dbuf) Zos_DbufDelete(_dbuf); \
} while (0)

/* test SIP create SIP message memory buffer */
#define TSIP_CREATE_MEMBUF(_membuf) do { \
    _membuf = TSIP_DBUF_CREATE(ZDBUF_TYPE_STRUCT, TSIP_DFT_MEMBUF_SIZE); \
} while (0)

/* test SIP message copy structure string */
#define TSIP_CPY_SSTR(_membuf, _dstsstr, _srcsstr) do { \
    if (Zos_SStrSCpy(_membuf, _dstsstr, _srcsstr) != ZOK) \
        return ZFAILED; \
} while (0)

/* test SIP message copy string with length */
#define TSIP_CPY_NSTR(_membuf, _dstsstr, _srcstr, _srclen) do { \
    if (Zos_SStrNCpy(_membuf, _dstsstr, _srcstr, _srclen) != ZOK) \
        return ZFAILED; \
} while (0)
```

### 5.3 Test SIP interfaces

```

/* test cal info get id */
ZINT Tsip_CallInfoGetID(ST_TSIP_CALL_INFO *pstCallInfo,
                       ZULONG *pdwSessUserId, ZULONG *pdwSessId, ZULONG *pdwDlgId,
                       ZULONG *pdwTransId, ST_SIP_TPT_ADDR *pstTptAddr)
{
    if (pstCallInfo == ZNULL)
        return ZFAILED;

    if (pdwSessUserId) *pdwSessUserId = pstCallInfo->dwUserId;
    if (pdwSessId) *pdwSessId = pstCallInfo->dwSessId;
    if (pdwDlgId) *pdwDlgId = pstCallInfo->dwDlgId;
    if (pdwTransId) *pdwTransId = pstCallInfo->dwTransId;

    if (pstTptAddr)
    {
        /* set remote address */
        pstTptAddr->ucType = (ZUCHAR)TSIP_STACK_PROTO_RUNNING;
        pstTptAddr->stLocalAddr.ucType = ZINET_IPV4;
        pstTptAddr->stRmtAddr.ucType = ZINET_IPV4;
        Zos_InetAddr(TSIP_STACK_IP, &pstTptAddr->stLocalAddr.u.dwIp);
        Zos_InetAddr(TSIP_STACK_IP, &pstTptAddr->stRmtAddr.u.dwIp);
        pstTptAddr->stLocalAddr.wPort = TSIP_STACK_PORT_RUNNING;

        if (pstCallInfo->ucIsTester)
            pstTptAddr->stRmtAddr.wPort = TSIP_CALLTEST_PORT_RUNNING;
        else
            pstTptAddr->stRmtAddr.wPort = TSIP_STACK_PORT_RUNNING;
    }

    return ZOK;
}

/* test sip generate init message,
   which is INVITE or non-INVITE first message */
ZINT Tsip_GenInitMsg(ST_SIP_MSG **ppstMsg, ZUCHAR ucMethod,
                    ST_TSIP_CALL_INFO *pstCallInfo)
{
    ST_ZOS_INET_ADDR stLocalAddr, stRmtAddr;
    ZCHAR *pcLocalName, *pcRemoteName;
    ST_ZOS_SSTR stDomain, stUserInfo;
    ST_SIP_HDR_FROM_TO *pstFrom;
    ST_SIP_HDR_FROM_TO *pstTo;
    ST_ZOS_DBUF *pstMemBuf;
    ST_SIP_MSG *pstMsg;

```

```
if (pstMsg == ZNULL || pstCallInfo == ZNULL)
    return ZFAILED;

/* create request message */
Sip_MsgCreate(&pstMsg);

/* get the memory buffer */
pstMemBuf = pstMsg->pstMemBuf;

if (ucMethod == EN_SIP_METHOD_REGISTER)
{
    ZOS_SSTR_SETS(&stDomain, "juphoon.com");

    /* fill request header */
    Sip_MsgFillReqLineByName(pstMemBuf, pstMsg, ucMethod, ZNULL,
                             &stDomain, 0);

    /* fill from header */
    pstFrom = Sip_CreateMsgHdr(pstMsg, EN_SIP_HDR_FROM);

    ZOS_SSTR_SETS(&stUserInfo, TSIP_SUA_NAME);
    Sip_HdrFillFromToByName(pstMemBuf, pstFrom, ZNULL, &stUserInfo,
                             &stDomain, 0, ZNULL);

    /* fill to header */
    pstTo = Sip_CreateMsgHdr(pstMsg, EN_SIP_HDR_TO);

    Sip_HdrFillFromToByName(pstMemBuf, pstTo, ZNULL, &stUserInfo,
                             &stDomain, 0, ZNULL);
}
else
{
    /* get local and remote name, ip, port */
    if (pstCallInfo->bUseCtmName == ZFALSE)
    {
        pcLocalName = TSIP_SUA_NAME;
        pcRemoteName = TSIP_TESTER_NAME;

        /* set local address */
        stLocalAddr.ucType = ZINET_IPV4;
        Zos_InetAddr(TSIP_STACK_IP, &stLocalAddr.u.dwIp);
        stLocalAddr.wPort = TSIP_STACK_PORT_RUNNING;
    }
}
```

```

        ZOS_INET_ADDR_COPY(&stRmtAddr, &pstCallInfo->stRmtAddr);
    }
    else
    {
        pcLocalName = pstCallInfo->stLocalSet.pcName;
        pcRemoteName = pstCallInfo->pstRemoteSet->pcName;

        /* set local address */
        stLocalAddr.ucType = ZINET_IPV4;
        Zos_InetAddr(pstCallInfo->stLocalSet.pcIp, &stLocalAddr.u.dwIp);
        stLocalAddr.wPort = pstCallInfo->stLocalSet.wPort;

        stRmtAddr.ucType = ZINET_IPV4;
        Zos_InetAddr(pstCallInfo->pstRemoteSet->pcIp, &stRmtAddr.u.dwIp);
        stRmtAddr.wPort = pstCallInfo->pstRemoteSet->wPort;
    }

    /* fill request header */
    ZOS_SSTR_SETS(&stUserInfo, pcRemoteName);
    Sip_MsgFillReqLineByIp(pstMemBuf, pstMsg, ucMethod, &stUserInfo,
                          &stRmtAddr);

    /* fill from header */
    pstFrom = Sip_CreateMsgHdr(pstMsg, EN_SIP_HDR_FROM);

    ZOS_SSTR_SETS(&stUserInfo, pcLocalName);
    Sip_HdrFillFromToByIp(pstMemBuf, pstFrom, ZNULL, &stUserInfo,
                          &stLocalAddr, ZNULL) ;

    /* fill to header */
    pstTo = Sip_CreateMsgHdr(pstMsg, EN_SIP_HDR_TO);

    stUserInfo.pcStr = pcRemoteName;
    stUserInfo.wLen = Zos_StrLen(pcRemoteName);
    Sip_HdrFillFromToByIp(pstMemBuf, pstTo, ZNULL, &stUserInfo,
                          &stRmtAddr, ZNULL)
}

*ppstMsg = pstMsg;

return ZOK;
}

/* test SIP generate PRACK message */

```

```
ZINT Tsip_GenPrackMsg(ST_SIP_MSG **ppPrack)
{
    ST_ZOS_DBUF *pstMemBuf;
    ST_SIP_MSG *pstPrack;
    ZULONG dwIp;

    /* create request message */
    Sip_MsgCreate(&pstPrack);

    /* get the memory buffer */
    pstMemBuf = pstMsg->pstMemBuf;

    /* fill request header */
    ZOS_SSTR_SETS(&stUesrInfo, TSIP_TESTER_NAME);

    stAddr.ucType = ZINET_IPV4;
    Zos_InetAddr(TSIP_CALL_TPT_IP, &stAddr.u.dwIp);
    stAddr.wPort = TSIP_CALL_TPT_UDP_PORT;

    /* fill request header */
    Sip_MsgFillReqLine(pstMemBuf, pstPrack, EN_SIP_METHOD_PRACK,
                       TSIP_TESTER_NAME, dwIp, TSIP_CALL_TPT_SERV1_PORT);

    *ppPrack = pstPrack;

    return ZOK;
}
```

## 5.4 Sip\_DecodeMsg

In the example below, function **Tsip\_DecodeMsg** has an input parameters **pstData** pointing to the data buffer where the message is stored. The message in the data buffer is going to be decoded by ABNF. Function **Tsip\_DecodeMsg** also has an output parameters **ppstMsg** pointing to the pointer which points to the decoded message. Function **Tsip\_DecodeMsg** calls function **Sip\_DecodeMsg** to decode a message.

```
/* test SIP decode message */
ZINT Tsip_DecodeMsg(ST_ZOS_DBUF *pstData, ST\_SIP\_MSG **ppstMsg)
{
    ST\_ABNF\_MSG stAbnfMsg;
    ST_ZOS_SSTR stDataStr;
    ST_ZOS_DBUF *pstMemBuf;
    ST\_SIP\_MSG *pstMsg;

    /* set the message info */
    Zos_DbufO2D(pstData, 0, &stDataStr.pcStr);
    stDataStr.wLen = (ZUSHORT)pstData->dwBufLen;

    /* allocate the dbuf */
    TSIP_CREATE_MEMBUF(pstMemBuf);

    /* add dbuffer into garbage bin */
    Zos_GabAddMem(pstGab, pstMemBuf, Tsip_DbufDelete);

    /* allocate memory for SIP message */
    pstMsg = TSIP_DBUF_ALLOC_CLRD(pstMemBuf, sizeof(ST_SIP_MSG));

    /* init the abnf message */
    Abnf_MsgInit(&stAbnfMsg, ZPROTOCOL_SIP, &stDataStr,
                pstMemBuf, ZNULL, ZTRUE);

    /* decode message */
    if (Sip_DecodeMsg(&stAbnfMsg, pstMsg) != ZOK)
    {
        TSIP_DELETE_MEMBUF(pstMemBuf)
        return ZFAILED;
    }

    /* clone message buffer */
    pstMsg->pstMsgBuf = pstData;
    *ppstMsg = pstMsg;

    return ZOK;
}
```

## 5.5 Sip\_EncodeMsg

```

/* SIP stack information */
#define SIP_STACK_IP      "192.168.0.1"
#define SIP_STACK_PORT   5060

/* test SIP send message by call transport */
ZINT Tsip_TptSendMsg(ZULONG dwServNo, ST SIP MSG *pstMsg)
{
    ST_ZOS_INET_ADDR stPeerAddr;
    ST_ABNF_MSG stAbnfMsg;
    ST_ZOS_DBUF *pstMsgbuf;

    if (pstMsg->pstMemBuf == ZNULL)
        return ZFAILED;

    /* create message buffer */
    pstMsgbuf = Zos_DbufCreate(ZNULL, ZDBUF_TYPE_BYTE, 1024);

    /* init the abnf message */
    Abnf_MsgInit(&stAbnfMsg, ZPROTOCOL_SIP, ZNULL,
                pstMsgbuf, ZNULL, ZFALSE)

    /* encode message */
    Sip_EncodeMsg(&stAbnfMsg, pstMsg);

    /* set stack address */
    stPeerAddr.ucType = ZINET_IPV4;
    Zos_InetAddr(TSIP_STACK_IP, &stPeerAddr.u.dwIp);
    stPeerAddr.wPort = TSIP_STACK_PORT;

    /* use tpt send data */
    Call_TptDataReq(dwServNo, &stPeerAddr, pstMsg->pstMsgBuf);

    TSIP_LOG_INFO((TSIP_LOGID, "tpt send data(%ld byte) ok.",
                  pstMsgbuf->dwBufLen));

    /* delete dbuf */
    TSIP_DELETE_MEMBUF(pstMsgbuf);

    return ZOK;
}

```

## 5.6 Sip\_SendSimReq

```

/* test sip send sim request */

```

```
ZINT Tsip_LiSimReq(ST_TSIP_CALL_INFO *pstCallInfo)
{
    ST_SIP_TPT_ADDR stTptAddr;
    ZULONG dwSessUserId;
    ST_SIP_MSG *pstMsg;

    pstCallInfo->dwSessId = ZMAXULONG;
    pstCallInfo->dwDlgId = ZMAXULONG;
    pstCallInfo->dwTransId = ZMAXULONG;

    /* generate invite message */
    Tsip_GenInitMsg(&pstMsg, EN_SIP_METHOD_INVITE, pstCallInfo);

    /* get the id from call info */
    Tsip_CallInfoGetID(pstCallInfo, &dwSessUserId, ZNULL, ZNULL, ZNULL,
        &stTptAddr);

    /* send the sim request */
    Sip_SendSimReq(pstCallInfo->dwUserId, ZMAXULONG, ZMAXULONG,
        &stTptAddr, pstMsg);

    return ZOK;
}
```

[References]

[Sip\\_SendSimRsp](#)

[Sip\\_SendSsmReq](#)

[Sip\\_SendSamReq](#)

[Sip\\_SendScmReq](#)

[Sip\\_SendSmmReq](#)

[Sip\\_SendStmReq](#)

[Sip\\_SendDamReq](#)

[Sip\\_SendCimReq](#)

## 5.7 Sip\_SendSimRsp

```
/* test sip send INVITE response */
ZINT Tsip_LiSimRsp(ST_TSIP_CALL_INFO *pstCallInfo)
{
    ZULONG dwSessUserId, dwSessId, dwDlgId, dwTransId;

    /* get the id from call info */
    Tsip_CallInfoGetID(pstCallInfo, &dwSessUserId, &dwSessId, &dwDlgId,
        &dwTransId, ZNULL);

    /* send sim request */
    Sip_Sip_SendSimRsp(dwSessUserId, dwSessId, ZMAXULONG, dwDlgId,
        ZMAXULONG, dwTransId, pstCallInfo->dwRspCode, ZNULL);

    return ZOK;
}
```

[References]

[Sip\\_SendSimReq](#)

[Sip\\_SendSsmRsp](#)

[Sip\\_SendSmmRsp](#)

[Sip\\_SendDamRsp](#)

[Sip\\_SendCimRsp](#)

## 5.8 Sip\_SendSsmReq

In the example, function **Tsip\_LiSsmReq** calls **Sip\_SendSsmReq** to send a session status request. Before sending any requests, there are several steps need to be taken. First, **Sip\_FindMsgHdr** is called to find the Cseq header and Rseq header by header type. Second, a PRACK message is generated by **Tsip\_GenPrackMsg**. Then, **Sip\_CreateMsgHdr** creates the rack header. The last step is to set the rack header. After all these steps are finished, **Sip\_SendSsmReq** is called to send the request.

```
/* test sip send ssm request */
ZINT Tsim_LiSsmReq(ST_TSIP_CALL_INFO *pstCallInfo)
{
    ZULONG dwSessUserId, dwSessId, dwDlgId, dwTransId;
    ST_SIP_HDR_RACK *pstRack;
    ST_SIP_HDR_RSEQ *pstRseq;
    ST_SIP_HDR_CSEQ *pstCseq;
    ST_SIP_MSG *pstMsg;

    /* find cseq header */
    pstCseq = Sip_FindMsgHdr(pstCallInfo->pstLastMsg, EN_SIP_HDR_CSEQ);

    /* find rseq header */
    pstRseq = Sip_FindMsgHdr(pstCallInfo->pstLastMsg, EN_SIP_HDR_RSEQ);

    /* generate prack message */
    Tsim_GenPrackMsg(&pstMsg);

    /* session event init */
    Tsim_SessEvtInit(pstEvt, pstEvt->pstMsg, pstCallInfo);

    /****** add RACK header *****/
    /* create rack header */
    pstRack = Sip_CreateMsgHdr(pstEvt->pstMsg, EN_SIP_HDR_RACK);

    /* set rack header */
    Sip_MsgFillHdrRack(pstEvt->pstMsg, pstCseq->dwCseqVal,
        pstRseq->dwRspNum, pstCseq->stMethod.ucType);

    /* get the id from call info */
    Tsim_CallInfoGetID(pstCallInfo, &dwSessUserId, &dwSessId, &dwDlgId,
        &dwTransId, &stTptAddr);

    /* send ssm request */
    Sip_SendSsmReq(dwSessUserId, dwSessId, ZMAXULONG, dwDlgId, ZMAXULONG,
        &stTptAddr, pstCallInfo->ucMethodType, pstMsg);

    return ZOK;
}
```

**[References]**

[Sip\\_SendSimReq](#)

## 5.9 Sip\_SendSamReq

```
/* test SIP send ACK request */
ZINT Tsip_LiSamReq(ST_TSIP_CALL_INFO *pstCallInfo)
{
    ZULONG dwSessUserId, dwSessId, dwDlgId;
    ST_SIP_TPT_ADDR stTptAddr;

    /* get the id from call info */
    Tsip_CallInfoGetID(pstCallInfo, &dwSessUserId, &dwSessId, &dwDlgId,
        ZNULL, &stTptAddr);

    /* send sam request */
    Sip_SendSamReq(dwSessUserId, dwSessId, ZMAXULONG, dwDlgId,
        ZMAXULONG, &stTptAddr, ZNULL);

    return ZOK;
}
```

## 5.10 Sip\_SendScmReq

```
/* test SIP send CANCEL request */
ZINT Tsip_LiScmReq(ST_TSIP_CALL_INFO *pstCallInfo)
{
    ZULONG dwSessUserId, dwSessId, dwDlgId;
    ST_SIP_TPT_ADDR stTptAddr;

    /* reset the trans id */
    pstCallInfo->dwTransId = ZMAXULONG;

    /* get the id from call info */
    Tsip_CallInfoGetID(pstCallInfo, &dwSessUserId, &dwSessId, &dwDlgId,
        ZNULL, &stTptAddr);

    /* send scm request */
    Sip_SendScmReq(dwSessUserId, dwSessId, ZMAXULONG, dwDlgId,
        ZMAXULONG, &stTptAddr, ZNULL);

    return ZOK;
}
```

### 5.11 Sip\_SendSmmReq

```
/* test SIP send re-INVITE request */
ZINT Tsip_LiSmmReq(ST_TSIP_CALL_INFO *pstCallInfo)
{
    ZULONG dwSessUserId, dwSessId, dwDlgId;
    ST_SIP_TPT_ADDR stTptAddr;

    /* get the id from call info */
    Tsip_CallInfoGetID(pstCallInfo, &dwSessUserId, &dwSessId, &dwDlgId,
        ZNULL, &stTptAddr);

    /* send smm request */
    Sip_SendSmmReq(dwSessUserId, dwSessId, ZMAXULONG, dwDlgId,
        ZMAXULONG, &stTptAddr, ZNULL);

    return ZOK;
}
```

### 5.12 Sip\_SendSmmRsp

```
/* test SIP send re-INVITE response */
ZINT Tsip_LiSmmRsp(ST_TSIP_CALL_INFO *pstCallInfo)
{
    ZULONG dwSessUserId, dwSessId, dwDlgId, dwTransId;

    /* get the id from call info */
    Tsip_CallInfoGetID(pstCallInfo, &dwSessUserId, &dwSessId, &dwDlgId,
        &dwTransId, ZNULL);

    /* send sim request */
    Sip_SendSmmRsp(dwSessUserId, dwSessId, ZMAXULONG, dwDlgId,
        ZMAXULONG, dwTransId, pstCallInfo->dwRspCode,
        ZNULL);

    return ZOK;
}
```

### 5.13 Sip\_SendStmReq

```
/* test SIP send BYE request */
ZINT Tsip_LiStmReq(ST_TSIP_CALL_INFO *pstCallInfo)
{
    ZULONG dwSessUserId, dwSessId, dwDlgId;
    ST_SIP_TPT_ADDR stTptAddr;

    /* reset the trans id */
    pstCallInfo->dwTransId = ZMAXULONG;

    /* get the id from call info */
    Tsip_CallInfoGetID(pstCallInfo, &dwSessUserId, &dwSessId, &dwDlgId,
        ZNULL, &stTptAddr);

    /* send stm request */
    Sip_SendStmReq(dwSessUserId, dwSessId, ZMAXULONG, dwDlgId,
        ZMAXULONG, &stTptAddr, ZNULL);

    return ZOK;

    return ZOK;
}
```

## 5.14 Sip\_SendDamReq

```

/* test sip send dam request */
ZINT Tsip_LiDamReq(ST_TSIP_CALL_INFO *pstCallInfo)
{
    ZULONG dwSessUserId, dwSessId, dwDlgId, dwTransId;
    ST_SIP_HDR_EVNT *pstHdrEvnt;
    ST_SIP_TPT_ADDR stTptAddr;
    ST_SIP_MSG *pstMsg;

    if (pstCallInfo->ucMethodType != EN_SIP_METHOD_NOTIFY)
    {
        /* generate non-invite message */
        Tsip_GenInitMsg(&pstMsg, pstCallInfo->ucMethodType,
                       PstCallInfo);
    }
    else /* NOTIFY */
    {
        /* create request message */
        Sip_MsgCreate(&pstMsg);
        pstMsg->ucPres = ZTRUE;

        Sip_MsgFillHdrSubsSta(pstMsg, pstCallInfo->ucSubsSta);
    }

    pstCallInfo->dwTransId = ZMAXULONG;

    /* if SUBSCRIBE method, set event package */
    if (pstCallInfo->ucMethodType == EN_SIP_METHOD_SUBS)
    {
        if (pstCallInfo->stEvent.wLen != 0)
        {
            pstHdrEvnt = Sip_CreateMsgHdr(pstEvnt->pstMsg, EN_SIP_HDR_EVNT);
            pstHdrEvnt->stEvntType.stEvntPkg.ucPres = ZTRUE;
            pstHdrEvnt->stEvntType.stEvntPkg.ucType = EN_SIP_EVNT_PKG_OTHER;
            TSIP_COPY_SSTR(pstMsg->pstMemBuf,
                          &pstHdrEvnt->stEvntType.stEvntPkg.stOther,
                          &pstCallInfo->stEvent);

            pstHdrEvnt->ucPres = ZTRUE;
        }

        /* get the id from call info */
        Tsip_CallInfoGetID(pstCallInfo, &dwSessUserId, &dwSessId, &dwDlgId,
                          &dwTransId, &stTptAddr);
    }
}

```

```
/* send dam request */
Sip_SendDamReq(dwSessUserId, dwSessId, ZMAXULONG, dwDlgId, ZMAXULONG,
               &stTptAddr, pstCallInfo->ucMethodType, pstMsg);

return ZOK;
}
```

## 5.15 Sip\_SendCimReq

```
/* test SIP send cim request */
ZINT Tsip_LiCimReq(ST_TSIP_CALL_INFO *pstCallInfo)
{
    ST_SIP_TPT_ADDR stTptAddr;
    ZULONG dwSessUserId;
    ST_SIP_MSG *pstMsg;

    /* generate no invite message */
    Tsip_GenInitMsg(&pstMsg, pstCallInfo->ucMethodType, pstCallInfo);

    pstCallInfo->dwSessId = ZMAXULONG;
    pstCallInfo->dwDlgId = ZMAXULONG;
    pstCallInfo->dwTransId = ZMAXULONG;

    /* get the id from call info */
    Tsip_CallInfoGetID(pstCallInfo, &dwSessUserId, ZNULL, ZNULL, ZNULL,
                       &stTptAddr);

    /* send cim request */
    Sip_SendCimReq(dwSessUserId, ZMAXULONG, ZMAXULONG, &stTptAddr,
                  pstCallInfo->ucMethodType, pstMsg);

    return ZOK;
}
```

## 5.16 Sip\_SendCimRsp

```
/* test SIP send cim response */
ZINT Tsip_LiCimRsp(ST_TSIP_CALL_INFO *pstCallInfo)
{
    ZULONG dwSessUserId, dwSessId, dwDlgId, dwTransId;

    /* get the id from call info */
    Tsip_CallInfoGetID(pstCallInfo, &dwSessUserId, &dwSessId, &dwDlgId,
        &dwTransId, ZNULL);

    /* send cim response */
    Sip_SendCimRsp(dwSessUserId, dwSessId, ZMAXULONG, dwDlgId,
        ZMAXULONG, dwTransId, pstCallInfo->ucMethodType,
        pstCallInfo->dwRspCode, pstRspMsg);

    return ZOK;
}
```